AD-A107 879 SAI COMSYSTEMS CORP MCLEAN VA F/G 9/2
SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES.(U)
JUN 77 T DENIKE, A HOLLAND, T WARD, H DESAI DAHC26-76-D-1004
UNCLASSIFIED USACSC-AT-77-11 NL

1 OF 3
AD
A107879

TECHNICAL DOCUMENTARY REPORT

U.S. ARMY COMPUTER SYSTEMS COMMAND

19

16 USACSC-AT-77-11

6

SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES

9 FINAL REPORT

Authors: T. Denike
A. Holland
T. Ward
H. Desai

10

11 3∅ JUN[redacted]77

12 214

PREPARED FOR:

U.S. ARMY COMPUTER SYSTEMS COMMAND
FORT BELVOIR, VIRGINIA 22060

PREPARED BY:

SAI COMSYSTEMS CORPORATION
MCLEAN, VIRGINIA 22101

15 CONTRACT DAHC26-76-D-1004

DISTRIBUTION STATEMENT:

APPROVED FOR PUBLIC RELEASE. DISTRIBUTION UNLIMITED.

392218

81 11 24 051

ABSTRACT

This document contains the results of a modification to the Software Portability Study, Delivery Order Number 9 to Contract DAHC26-D-1004. This study concentrated on determining the procedures required to convert software systems written in COBOL in accordance with USACSC standards to a portable COBOL, Florida 74. A further conversion from the Portable Standard COBOL (PSC) to a COBOL executable on the Digital Equipment Corporation (CDEC) PDP 11/70 minicomputer was studied and is presented.

FORWARD

This document was prepared under the authority of USACSC Contract Number DAHC26-76-D-1004, and was prepared by SAI Comsystems for the U.S. Army Computer System Command. This study reports the COBOL software study.

DISCLAIMER

DISPOSITION INSTRUCTIONS

TABLE OF CONTENTS

CHAPTER I

OVERVIEW

1.1 INTRODUCTION

The United States Army Computer Systems Command (USACSC) has for many years been developing and maintaining software systems for use throughout the army. The majority of these software systems have been written in COBOL and executed on IBM 360 systems. Due to the rapid advances in computer hardware, the competitive nature of the computer industry and federal government computer procurement practices, it is reasonable to expect that the present software systems will be required to be executed on hardware for which they were not developed. With this in mind, a greater emphasis is being placed on software transportability within the USACSC.

1.2 PORTABILITY

1.2.1 By definition, software portability implies the degree of executability of a high-level language program in multiple and/or varied computer environments. That is, if a program is executable in a foreign environment from which it was developed with minimal or no modification, it is considered to be portable; otherwise, the program is not portable.

1.2.2 Program portability involves many aspects of data processing.

Briefly, to cite a few:

- Compatability among computer vendors,
- Compiler compatability for a given high-level language.
- Compiler compatability within the same vendor and/or other vendors.
- Compatability of a given high-level language as used in various computer environments.
- Program application.
- Determination of the degree to which a program is or is not portable. (That is, if modification is required, how much modification is too much; how many special cases should be incorporated into considerations for portability).
- Determination of a universally executable instruction subset of a high-level language.
- Program dependency on the computer environment.
- Degree of program interdependence with Job Control Language.
- Degree of programmed system dependent device specification.
- Variability in special features such as internal sorts, internal merges, CALL's, etc.
- Compatibility between operating systems within a vendor (e.g., IBM DOS, OS, VS, HASP, MVT, etc) and with other vendors and their variations.
- Compatability in system data management procedures (within a given vendor or given vendor to another vendor.
- Compatability in the relative intelligence built into the original versus the recipient environment (the amount of programmed

information which is or is not required by the program depending on the computer environment).

1.3 PORTABILITY STUDY

1.3.1 The USACSC has tasked SAI Comsystems Corporation as Delivery Order Number 9 of Contract DAHC26-76-D-1004 to study the question of portability in the context of the command environment. The final document, Software Portability Study - Volumes I and II was delivered on April 15, 1977. In the course of this study, the following items were considered in the form of comparisons-similarities and dissimilarities.

- COBOL - Programming language study.
- Job Control Language (JCL) study.
- Executive Software Study (as applicable) in the domain of Operating Systems Environment.
- Computer Hardware Study.

This study was conducted using USACSC minimum hardware configuration requirements as the norm. All software comparisons are based on current versions of the language, JCL, and Executive Software as used by USACSC. The vendors considered include: Burroughs 3500, CDC 3300, 6600 and 7600 Series, Honeywell 6000 Series, Univac 1100 Series, Data General Eclipse, Digital Equipment Corp. PDP 11, and Interdata 8/32.

Also included in the language (COBOL) study are USACSC COBOL and ANSI '74 COBOL.

The purpose of this extension is to take USACAC COBOL and constrain it to become portable so that the portable COBOL is usable by the PDP 11 - hardware environment.

1.4 DOCUMENT STRUCTURE

Chapter 2 of this document presents the general scope of the study. Chapter 3 presents the methodology of converting USACSC COBOL to Portable Standard COBOL (PSC) (Ref: Programming Procedures Manual USACSC 18-1-1; Optimal COBOL Subset for Software Portability DAAG29-77-G-0058) as well as the PSC to PDP 11 COBOL (Ref: Optimal COBOL Subset for Software Portability DAAG29-77-G-0058); PDP-11 COBOL User's Guide No. DEC-11-LCUGA-B-D). Appendix A consists of a heirarchy chart and the detail conversion process for USACSC COBOL to PSC. Appendix B consists of a heirarchy chart and the detail conversion process for PSC to PDP 11 COBOL.

CHAPTER 2

STUDY SCOPE

2.1 INTRODUCTION

This chapter presents the scope of the modification to the original Software Portability Study delivery order and some general understandings and concepts required to complete the task.

2.2 TASK MODIFICATION

Where the portability study included many aspects of the portability question, the modification concentrated on the methodology of achieving the conversion of systems written in COBOL. Specifically, the conversion of systems written in accordance with USACSC standard to an intermediate COBOL language and thence to a specific hardware dependent COBOL, PDP 11, is being determined.

2.3 BACKGROUND

2.3.1 The methodology being evaluated and presented in this document is general in nature and applicable to both manual and automated modes of operation. The stress has, however, been placed on an automated mode of operation where problem areas are noted and manual intervention is required.

2.3.2 Due to the complexity of converting COBOL systems from one hardware vendor to another, it is evident that no automated mode of conversion will be self-sufficient. Too many inconsistancies occur between vendors implemented COBOL even though the vendors indicate that the language implemented is in accordance with an ANSI standard. These inconstancies appear in two forms:

- The entire set of ANSI COBOL is not implemented.
- Vendor extensions have been implemented to fully utilize the unique features of the vendor hardware.

2.3.3 As indicated in the portability study report, the strict adherence to programming systems as a minimal subset (i.e., COBOL statements implemented by all vendors) to achieve portability, too severely restricts the programmer in capability and makes systems thus programmed inefficient on all hardware systems on which it is executed. Therefore, although the portable COBOL in which systems are programmed and maintained may be a minimal subset, the translation from this language to any hardware dependent COBOL must attempt to transform the portable subset into the maximum vendor subset possible to obtain maximum efficiency. The trade off in this philosophy is that the greater the vendor subset attempted, then the greater the complexity of the translator.

2.4 TRANSLATOR TRADE-OFF

2.4.1 The critical point in this trade-off is when the cost of developing the translator exceeds the expected level of manual intervention in modifying the code based on error messages. The translator defined as a result of this effort has been designed with an attempt to minimize the translator development cost and manual intervention and yet enable the utilization of a maximum target COBOL subset.

2.4.2 The scope of this extension is summarized in the block diagram (Figure 2.1). The approach is to achieve portability in two stages. The first stage will accomplish the conversion of USACSC COBOL to an intermediate

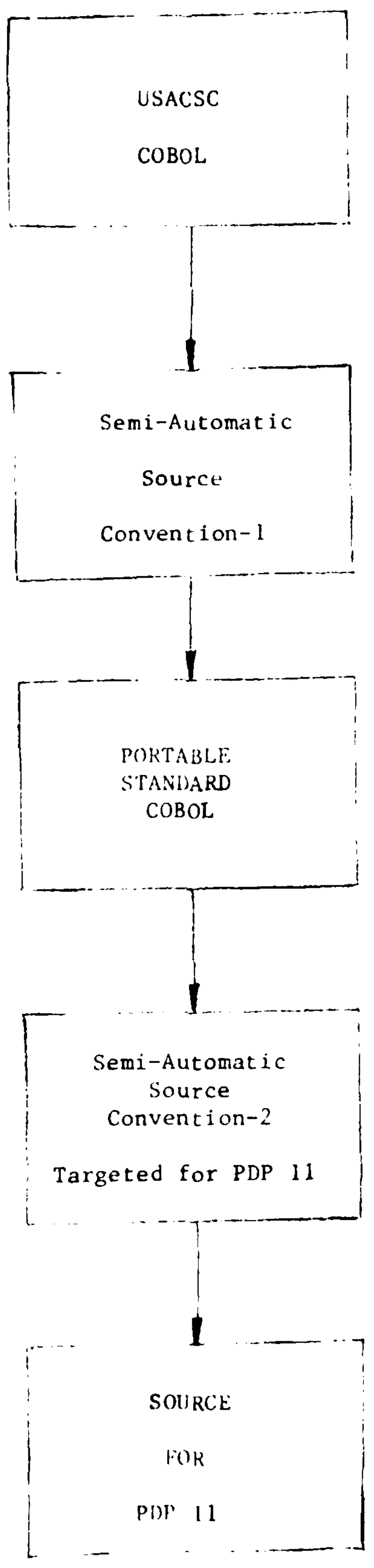


Figure 2.1 BLOCK DIAGRAM

COBOL language element set. This COBOL set, as recommended by University of Florida and agreed upon by USACSC, is a subset of ANSI '74 COBOL. Throughout the study, this intermediate COBOL will be referred to as Portable Standard COBOL (PDS). Also, PDS, being the proper subset (every element in the subset is present in the set) of ANSI '74, will be a usable COBOL on all the computers that support full ANSI '74 COBOL.

2.4.3 The second stage is to translate PDS to PDP 11 COBOL (or other desired hardware environment). This conversion has been accomplished keeping in mind the present state of the art for the COBOL available on PDP 11.

CHAPTER 3

CONVERSION PROCEDURES

3.1 INTRODUCTION

The primary function of these conversion procedures is to assist a programmer to achieve a conversion smoothly. Given a definite target COBOL environment, one can design a system that will allow a conversion. These conversion procedures serve dual purposes. It guides towards a design logic for conversion as well as lists many abnormalities that need to be resolved before automatic conversion can be achieved. This chapter discusses in detail the rationale and procedure for conversion of COBOL code maintained by USACSC to the Portable Standard COBOL (PSC) to PDP 11/70 COBOL. Also included is a discussion of Appendix A and B.

3.2 CONVERSION SCOPE

3.2.1 The conversion process is being presented strictly in terms of the COBOL language. The major consideration in this process is in terms of those capabilities of the compilers used to prepare the code for execution. JCL or Executive Software are identified only if a given function is not available in the target COBOL.

3.3 APPENDIX DESCRIPTION

3.3.1 Two appendicies are provided as part of this document. Appendix A presents the conversion process for USACSC to PSC . Appendix B presents the conversion process for PSC to PDP 11/70 COBOL. Each appendix consists of a heirarchial chart and a series of IPO's required for the conversion process.

3.3.2 The first figure in each appendix is a heirarchial representation of all the statements available for conversion from the source computer.

3.3.3 The numbering technique follows the heirarchy convention by levels, i.e.,

level 1 - n

level 2 - n.1; n.2, ...

level 3 - n.1.1; n.1.2, ...

level 4 - n.1.1.1; n.1.1.2, ...

and so on.

3.3.4 After page 1 in each appendix, there follows a set of detailed input-process-output (IPO). On every page of this IPO, one lowest level COBOL element conversion is described. All IPO's adhere to the same format. The input of IPO is a COBOL source (element). The process part of IPO describes the methodology and/or logic required to decide the outcome. The output section of IPO describes the final result of the translation process.

3.3.5 Throughout the IPO's, all ANSI '74 COBOL syntax conventions are used. The only liberty taken is the shading of an option of the input element to indicate that the option is a problem area in conversion. Also, for an element with multiple options, if some options are not directly transferrable, the output section indicates this by showing possible translations as described in the process section. Also, for the sake of completeness, one lowest level element may appear at more than one place in the heirarchy chart; however, they all point to the same detail IPO page.

3.3.6 The usefulness of IPO's became apparent during an attempt to convert a program from USACSC COBOL to PSC. The following is an example indicative of the method of using the IPO's. The programmer performing the conversion is assumed to be familiar with both source and target COBOL systems.

1. Find the IPO corresponding to the source statement. Use heirarchial charts by division as a quick reference.
2. Follow the procedure listed in the process section of the IPO.
3. If a warning message is indicated, several situations may have occurred.
    a) A portion of the source statement is omitted from the output as it is not required in the target COBOL. However, the omission must be accomplished in the JCL.
    b) A portion of the source statement is omitted - not required by the targeted environment.

3.4 RESTRICTIONS AND LIMITATIONS

During the entire study, only one area has not been discussed. This is the restrictions and limitations of the compilers for which these conversions are being specified. A thorough check of the target compiler together with practical investigation is a must at this stage. Some of these areas are:

1. Number of GO TO's available in a GO TO -- DEPENDING UPON clause.
2. Total number of nesting levels available in a given compiler.
3. Naming conventions.
4. Reserved word list.
5. Number of bits to a byte.
6. Word and boundry alignment.
7. Numeric storage formats.
8. Collating sequence.

3.5 SUMMARY

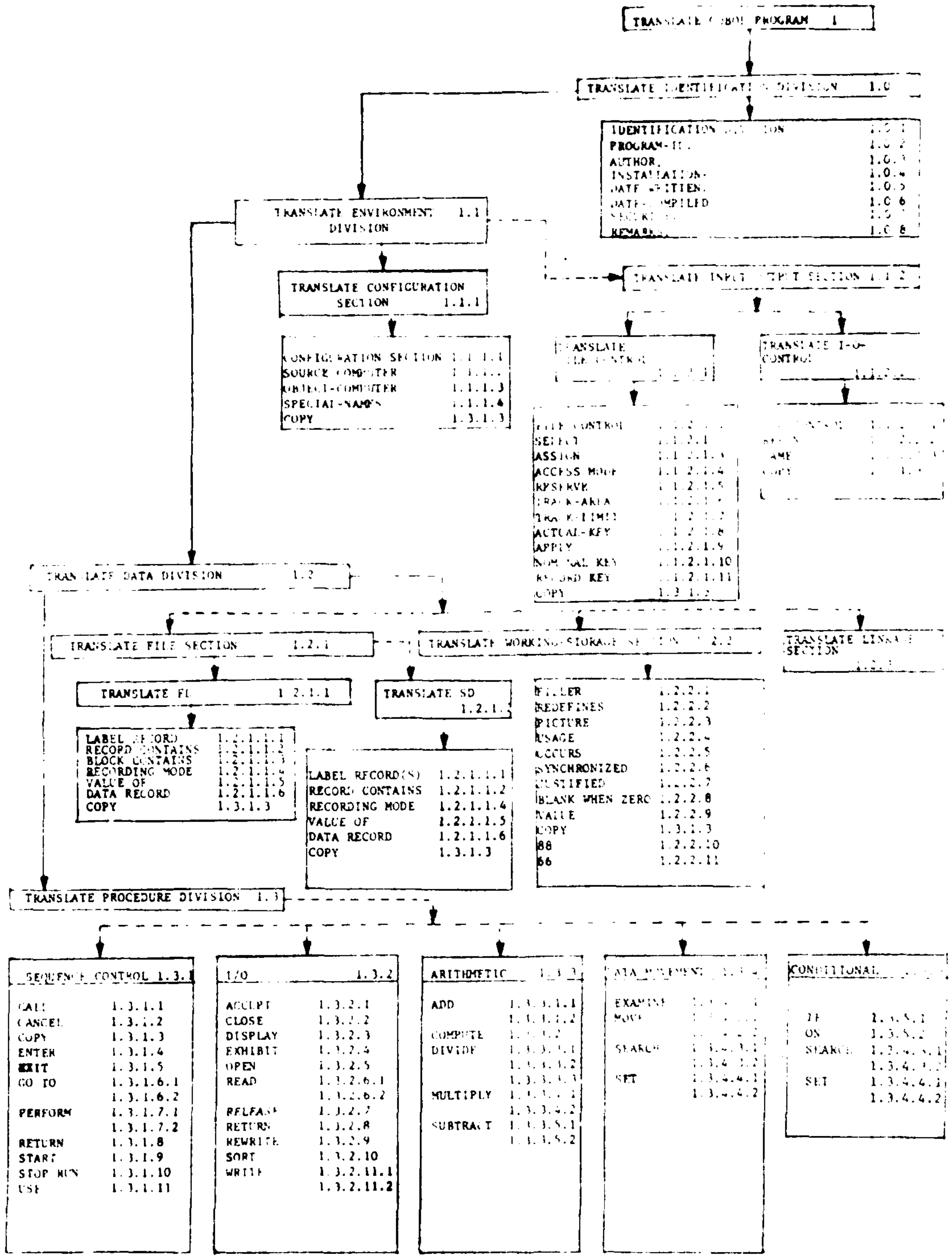
Even though this study covers all the aspects of COBOL conversion with respect to USACSC and PSC , this is by no means a 'Total' [illegible] conversion. One must also convert the data from the old environment to the new environment before a successful execution is possible. Also all the subroutines, utilities and any macros must be converted to the new environment before program execution. Finally, JCL conversion must also be accomplished. It is also likely JCL conversion might effect source code in the new environment.

APPENDIX A

USACSC COBOL

TO

PORTABLE STANDARD COBOL (PSC)


TRANSLATE COBOL PROGRAM 1
TRANSLATE IDENTIFICATION DIVISION 1.0
IDENTIFICATION DIVISION 1.0.1
PROGRAM-ID. 1.0.2
AUTHOR. 1.0.3
INSTALLATION. 1.0.4
DATE-WRITTEN. 1.0.5
DATE-COMPILED 1.0.6
REMARKS. 1.0.8
TRANSLATE ENVIRONMENT DIVISION 1.1
TRANSLATE CONFIGURATION SECTION 1.1.1
CONFIGURATION SECTION 1.1.1.1
SOURCE-COMPUTER
OBJECT-COMPUTER 1.1.1.3
SPECIAL-NAMES 1.1.1.4
COPY 1.3.1.3
TRANSLATE INPUT-OUTPUT SECTION 1.1.2
TRANSLATE FILE-CONTROL
TRANSLATE I-O-CONTROL
FILE-CONTROL
SELECT 1.1.2.1
ASSIGN
ACCESS MODE 1.1.2.1.4
RESERVE 1.1.2.1.5
TRACK-AREA
TRACK-LIMIT
ACTUAL-KEY 1.1.2.1.8
APPLY 1.1.2.1.9
NOMINAL KEY 1.1.2.1.10
RECORD KEY 1.1.2.1.11
COPY
TRANSLATE DATA DIVISION 1.2
TRANSLATE FILE SECTION 1.2.1
TRANSLATE WORKING-STORAGE SECTION 1.2.2
TRANSLATE LINKAGE SECTION
TRANSLATE FD 1.2.1.1
LABEL RECORD 1.2.1.1.1
RECORD CONTAINS 1.2.1.1.2
BLOCK CONTAINS 1.2.1.1.3
RECORDING MODE 1.2.1.1.4
VALUE OF 1.2.1.1.5
DATA RECORD 1.2.1.1.6
COPY 1.3.1.3
TRANSLATE SD 1.2.1.2
LABEL RECORD(S) 1.2.1.1.1
RECORD CONTAINS 1.2.1.1.2
RECORDING MODE 1.2.1.1.4
VALUE OF 1.2.1.1.5
DATA RECORD 1.2.1.1.6
COPY 1.3.1.3
FILLER 1.2.2.1
REDEFINES 1.2.2.2
PICTURE 1.2.2.3
USAGE 1.2.2.4
OCCURS 1.2.2.5
SYNCHRONIZED 1.2.2.6
JUSTIFIED 1.2.2.7
BLANK WHEN ZERO 1.2.2.8
VALUE 1.2.2.9
COPY 1.3.1.3
88 1.2.2.10
66 1.2.2.11
TRANSLATE PROCEDURE DIVISION 1.3
SEQUENCE CONTROL 1.3.1
CALL 1.3.1.1
CANCEL 1.3.1.2
COPY 1.3.1.3
ENTER 1.3.1.4
EXIT 1.3.1.5
GO TO 1.3.1.6.1
1.3.1.6.2
PERFORM 1.3.1.7.1
1.3.1.7.2
RETURN 1.3.1.8
START 1.3.1.9
STOP RUN 1.3.1.10
USE 1.3.1.11
I/O 1.3.2
ACCEPT 1.3.2.1
CLOSE 1.3.2.2
DISPLAY 1.3.2.3
EXHIBIT 1.3.2.4
OPEN 1.3.2.5
READ 1.3.2.6.1
1.3.2.6.2
RELEASE 1.3.2.7
RETURN 1.3.2.8
REWRITE 1.3.2.9
SORT 1.3.2.10
WRITE 1.3.2.11.1
1.3.2.11.2
ARITHMETIC 1.3.3
ADD 1.3.3.1.1
1.3.3.1.2
COMPUTE
DIVIDE 1.3.3.3.1
1.3.3.3.2
1.3.3.3.3
MULTIPLY
1.3.3.4.2
SUBTRACT 1.3.3.5.1
1.3.3.5.2
DATA MOVEMENT
EXAMINE
MOVE
SEARCH 1.3.4.3.1
1.3.4.3.2
SET 1.3.4.4.1
1.3.4.4.2
CONDITIONAL
IF
ON 1.3.5.2
SEARCH
SET 1.3.4.4.1
1.3.4.4.2

NO: 1.0.1

INPUT	PROCESS	OUTPUT
<u>IDENTIFICATION</u> <u>DIVISION</u>.	**Read entire statement, then copy to output area**	<u>IDENTIFICATION</u> <u>DIVISION</u>.

NO: 1.0.2

INPUT	PROCESS	OUTPUT
<u>PROGRAM-ID</u>. <u>program-name.</u>	Read entire statement, then copy to output area.	<u>PROGRAM-ID.</u> <u>program-name</u>.

NO: 1.[illegible].3

INPUT	PROCESS	OUTPUT
<u>AUTHOR</u>. [comment-entry] ...	Read entire statement, then copy to output area.	<u>AUTHOR</u>. [comment-entry] ...

NO: 1.0.4

INPUT	PROCESS	OUTPUT
INSTALLATION. [comment-entry] ...	Read entire statement, then copy to output area.	INSTALLATION. [comment-entry] ...

NO: 1.0.5

INPUT	PROCESS	OUTPUT
DATE-WRITTEN. [comment-entry] ...	Read entire statement, then copy to output area	DATE-WRITTEN. [comment-entry] ...

NO: 1.0.6

INPUT	PROCESS	OUTPUT
DATE-COMPILED. [comment-entry] ...	Read entire statement, then copy to output area.	DATE-COMPILED. [comment-entry] ...

NO: 1.0.7

INPUT	PROCESS	OUTPUT
[<u>SECURITY</u>. [comment-entry] ...]	Read entire statement, then copy to output area.	[<u>SECURITY</u>. [comment-entry] ...]

NO: 1.0.8

INPUT	PROCESS	OUTPUT
REMARKS. comment-entry	A. Read a statement B. Move an asterisk in column 7, and copy the entire statement. C. If a major division header is encountered, stop process. Else D. Move an asterisk to column 7 of every statement.	*REMARKS * comment-entry

NO: 1.1

INPUT	PROCESS	OUTPUT
ENVIRONMENT DIVISION.	Read entire statement, then copy to output area.	ENVIRONMENT DIVISION.

NO: 1.1.1.1

INPUT	PROCESS	OUTPUT
<u>CONFIGURATION</u> <u>SECTION</u>.	Read entire statement, then copy to output area.	<u>CONFIGURATION</u> <u>SECTION</u>.

NO: 1.1.1.2

INPUT	PROCESS	OUTPUT
<u>SOURCE-COMPUTER</u>. computer-name.	Read entire statement, then copy to output area.	<u>SOURCE-COMPUTER</u>. *computer-name*.

NO: 1.1.1.3

INPUT	PROCESS	OUTPUT
OBJECT-COMPUTER. computer name. [SEGMENT-LIMIT IS segment number] .	Read entire statement, then copy to output area.	OBJECT-COMPUTER. computer name. [SEGMENT-LIMIT IS segment number] .

NO: 1.1.1.4

INPUT	PROCESS	OUTPUT
[SPECIAL-NAMES. [implementor-name IS mnemonic name] ...]	Read Entire statement, then copy to output	[SPECIAL-NAMES. [implementor-name IS mnemonic name] ...]

NO: I.1.2

INPUT	PROCESS	OUTPUT
INPUT-OUTPUT SECTION.	Read entire statement, then copy to output area.	INPUT-OUTPUT SECTION.

NO: 2.1.2.1.2

INPUT	PROCESS	OUTPUT
<u>SELECT</u> file-name.	Read entire statement, then copy to output area.	<u>SELECT</u> file-name.

NO: 1.1.2.1.2

INPUT

SELECT [[redacted]]
file-name.

PROCESS

A. Read the entire statement.

B. If the OPTIONAL option is found, eliminate it from the statement; copy the statement to the output area, and write a warning message.

C. Else, copy the statement to the output area.

NOTE: Warning message should indicate that program execution may be affected.

OUTPUT

SELECT file-name
[**warning-message**]

NO: 1.1.2.1.3.

INPUT	PROCESS	OUTPUT
ASSIGN TO [integer] implementor-name-1 [,implementor-name-2] ... [redacted]	A. Read the entire statement. B. If the option is found [FOR MULTIPLE {REEL / UNIT}] ,eliminate it from the statement;copy the statement to the output area, and write a warning message. C. Else, copy the statement to the output area. NOTE: The warning message should indicate that program execution may be affected.	ASSIGN TO implementor-name-1 [,implementor-name-2] [**warning-message]

NO: 1.1.2.1.4

INPUT	PROCESS	OUTPUT
[ACCESS MODE IS] [{SEQUENTIAL / RANDOM}]	Read entire statement, then copy to output area.	[ACCESS MODE IS] [{SEQUENTIAL / RANDOM}]

NO: 1.1.2.1.5

INPUT	PROCESS	OUTPUT
RESERVE int [[redacted] $\left\{\begin{array}{l}\underline{\text{AREA}}\\ \underline{\text{AREAS}}\end{array}\right\}$]	A. Read entire statement B. Move the words "RESERVE" and "int" to output area. C. Check for the possibility of the verb "ALTERNATE". If this verb is existent, it must be eliminated. (NOTE: Elimination of verb may effect program execution; or there may exist in the operating system some equivalence to "ALTERNATE"). D. Move the remainder of the statement to the output area and add to "RESERVE int".	RESERVE int RESERVE int $\left\{\begin{array}{l}\underline{\text{AREA}}\\ \underline{\text{AREAS}}\end{array}\right\}$

NO: 1.1.2.1.6

INPUT	PROCESS	OUTPUT
<u>TRACK AREA</u> IS $\left\{\begin{array}{l}\text{data-name}\\ \text{int}\end{array}\right\}$ CHARACTERS	A. Read the entire statement. B. Since there is no-equivalence, the statement should be transferred to the output area with a warning message indicating no-equivalency. NOTE: There may be some effect on execution of the program by eliminating this statement.	TRACK statement ***WARNING message

NO: 1.1.2.1.7

INPUT	PROCESS	OUTPUT
<u>TRACK-LIMIT</u> IS int $\left\{\begin{matrix}\text{TRACK}\\\text{TRACKS}\end{matrix}\right\}$	A. Read the entire statement. B. Since there is no-equivalence, this statement should be transferred to the output area with a warning message indicating no-equivalency. NOTE: There may be some effect on the program by eliminating this statement.	TRACK statement ***WARNING message

NO: 1.1.2.1.8

INPUT	PROCESS	OUTPUT
ACTUAL **KEY** IS data-name	A. Read the entire statement. B. Since there is no-equivalent, this statement should be transferred to the output area with a warning message indicating no-equivalency. NOTE: The elimination of this statement may effect execution of this program, and the warning message should reflect this fact.	ACTUAL statement *** WARNING message

NO: 1.1.2.1.9

INPUT	PROCESS	OUTPUT
APPLY WRITE-ONLY on file-name-1 [file-name-2]	A. Read the entire statement. B. Since there is no equivalence, then the statement should be transferred with a warning message of no equivalency. Note: There might exist within the operating system or J.C.L. directives for write-only processing. This fact may be included in the warning message.	APPLY statement *** WARNING message

NO: 1.1.2.1.10

INPUT	PROCESS	OUTPUT
NOMINAL KEY IS data-name.	A. Read entire statement. B. Copy it with an asterisk, '*', in column seven. C. Write a warning message to indicate that the statement is not in ANSI COBOL. NOTE: Warning message should indicate that program execution may be affected.	COL. 7 * NOMINAL KEY IS data-name. ** WARNING - message

NO: 1.1.2.1.11

INPUT	PROCESS	OUTPUT
<u>RECORD</u> <u>KEY</u> IS data-name	Read the entire statement and copy it to the output area.	<u>RECORD</u> <u>KEY</u> IS data-name-1

NO: 1.1.2.2.1

INPUT	PROCESS	OUTPUT
<u>I-O-CONTROL</u>.	Read entire statement, then copy to output area.	<u>I-O-CONTROL</u>.

NO: 1.1.2.2.2

INPUT

```
[RERUN ON implementor-name
 EVERY { { END OF {REEL } }
         {        {UNIT } }
         { integer RECORDS }
 OF file-name } ] ...
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
[RERUN ON implementor-name
 EVERY { { END OF {REEL } }
         {        {UNIT } }
         { integer RECORDS }
 OF file name } ] ...
```

NO: 1.1.2.2.3

INPUT	PROCESS	OUTPUT
$\underline{\text{SAME}} \left\{ \begin{array}{l} \underline{\text{RECORD}} \\ \underline{\text{SAME}} \end{array} \right\}$ AREA FOR file-name-1 $\left\{ \text{file-name-2} \right\}$	A. Read entire statement. B. If RECORD or SORT option is used, statement should be moved to output area with warning message. C. Otherwise copy this statement to output area.	SAME statement ***WARNING message regarding RECORD or SORT SAME AREA FOR file-name-1 file-name-2

NO: 1.2

INPUT	PROCESS	OUTPUT
<u>DATA DIVISION</u>	Read entire statement, then copy to output area.	<u>DATA DIVISION</u>.

NO: 1.2.1

INPUT	PROCESS	OUTPUT
FILE SECTION.	Read entire statement, then copy to output area.	FILE SECTION.

NO: 1.2.1.1

INPUT	PROCESS	OUTPUT
FD file-name	Read entire statement, then copy to output area.	FD file-name

NO: 1.2.1.1.1

INPUT	PROCESS	OUTPUT
<u>LABEL RECORDS</u> ARE $\left\{\begin{array}{l}\underline{\text{STANDARD}}\\ \underline{\text{OMITTED}}\end{array}\right\}$	Read entire statement, then copy to output area.	<u>LABEL RECORDS</u> ARE $\left\{\begin{array}{l}\underline{\text{STANDARD}}\\ \underline{\text{OMITTED}}\end{array}\right\}$

NO: 1.2.1.1.2

INPUT	PROCESS	OUTPUT
[RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS]	Read entire statement, then copy to output area.	[RECORD CONTAINS [integer-1 TO] integer-2 CHARACTERS]

NO: 1.2.1.1.3

INPUT	PROCESS	OUTPUT
BLOCK CONTAINS [redacted] int-2 {RECORDS CHARACTERS}	A. Read entire statement. B. Move "BLOCK CONTAINS" verbs to the output area. C. Since the output code set doesn't allow a range capability, if the option "int-1 TO" must be eliminated. (NOTE: Elimination of this option may **not** effect execution). D. Add the remaining attributes to the output area to complete statement.	BLOCK CONTAINS BLOCK CONTAINS int-2 {RECORDS CHARACTERS}

NO: 1.2.1.1.4

INPUT	PROCESS	OUTPUT
RECORDING MODE IS $\left\{\begin{array}{c}\underline{F}\\\underline{U}\\\underline{V}\end{array}\right\}$	A. Read the entire statement. B. Move asterisk in column 7 and copy the option. C. Since there exists a no-equivalence condition, a warning message is printed. NOTE: There may exist within the operating system, and/or J.C.L., a means for indicating recording functions. If not, this statement's elimination may effect the results from execution of the program.	RECORDING statement. *** WARNING message.

INPUT

```
VALUE OF impiimentor-name-1

     IS     [redacted]
            literal-1

  implementor-name-2
            [redacted]
            literal-2
```

PROCESS

A. Read entire statement.

B. Move statement up to the "IS" verb to the output area.

C. Check if a data-name or literal has been used. If a data-name has been used, the statement should be moved to the output area with a warning message indicating no-equivalency. If data-name-0 has been defined previously, substitute literal-1 with appropriate value.

D. Else, add the literal field to the statement in the output area.

E. If additional fields are within the statement, the same test indicated in step C (above) should be adhered to, else the additional fields should be added to the output area.

OUTPUT

```
VALUE of implementor-name-1
       IS

VALUE clause

***WARNING message

VALUE OF implementor-name-1
     IS literal-1

VALUE OF implementor-name-1
     IS literal-1

  [implementor-name-2
       literal-2     ]
```

NO: 1.2.1.1.6

INPUT	PROCESS	OUTPUT
[DATA] {<u>RECORD</u> IS / <u>RECORDS</u> ARE} data-name-1 [,data-name-2] ...	Read entire statement, then copy to output area.	[DATA] {<u>RECORD</u> IS / <u>RECORDS</u> ARE} data-name-1 [,data-name-2] ...

NO: 1.2.2

INPUT	PROCESS	OUTPUT
<u>WORKING-STORAGE</u> <u>SECTION</u>.	Read entire statement, then copy to output area.	<u>WORKING-STORAGE</u> <u>SECTION</u>.

NO: 1.2.1.2

INPUT	PROCESS	OUTPUT
<u>SD</u> file-name	Read entire statement, then copy to output area.	<u>SD</u> file-name

NO: 1.2.2.1

INPUT	PROCESS	OUTPUT
level-number $\left\{\begin{array}{l}\text{data-name}\\ \underline{\text{FILLER}}\end{array}\right\}$	Read entire statement, then copy to output area.	level-number $\left\{\begin{array}{l}\text{data-name}\\ \underline{\text{FILLER}}\end{array}\right\}$

NO: 1.2.2.2

INPUT	PROCESS	OUTPUT
REDEFINES data-name	Read entire statement then copy to output area.	REDEFINES data-name

NO: 1.2.2.3

INPUT	PROCESS	OUTPUT
$\left\{\begin{array}{l}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{array}\right\}$ IS character-string	Read entire statement then copy to output area.	$\left\{\begin{array}{l}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{array}\right\}$ IS character-string

NO: 1.2.2.4

INPUT	PROCESS	OUTPUT
USAGE IS { COMPUTATIONAL / COMP / DISPLAY / INDEX / COMPUTATIONAL-3 / COMP-3 }	A. Read entire statement. B. If "COMPUTATIONAL-3" or "COMP-3" has been used, move statement to output area with warning message. (Note: Elimination of this phrase may effect execution.) C. Otherwise, copy the statement to the output area.	**WARNING statement on COMP-3, etc. USAGE IS { COMPUTATIONAL / COMP / DISPLAY / INDEX }

NO: 1.2.2.[illegible]

INPUT

```
OCCURS

  { integer-1 TIMES
  { integer-1 to
  { integer-2 TIMES

  DEPENDING ON data-name }

[ { ASCENDING  }  KEY IS
[ { DESCENDING }

  data-name-1  [ ,data-
  name-2 ] ...
[ INDEXED BY
  index-name-1  [ ,index-
  name-2 ] ...
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
OCCURS

  { integer-1 TIMES
  { integer-1 to
  { integer-2 TIMES

  DEPENDING ON data-name }

[ { ASCENDING  }  KEY IS
[ { DESCENDING }

  data-name-1  [ ,data
  name-2 ] ...
[ INDEXED BY
  index-name-1  [ ,index-
  name-2 ] ...
```

NO: 1.2.2.6

INPUT	PROCESS	OUTPUT
$\left[\begin{Bmatrix}\underline{\text{SYNCHRONIZED}}\\ \underline{\text{SYNC}}\end{Bmatrix}\begin{bmatrix}\underline{\text{LEFT}}\\ \underline{\text{RIGHT}}\end{bmatrix}\right]$	Read entire statement, then copy to output area.	$\left[\begin{Bmatrix}\underline{\text{SYNCHRONIZED}}\\ \underline{\text{SYNC}}\end{Bmatrix}\begin{bmatrix}\underline{\text{LEFT}}\\ \underline{\text{RIGHT}}\end{bmatrix}\right]$

NO: 1.2.2.7

INPUT	PROCESS	OUTPUT
$\left[\left\{\begin{array}{l}\underline{\text{JUSTIFIED}}\\ \underline{\text{JUST}}\end{array}\right\}\ \text{RIGHT}\right]$	Read entire statement, then copy to output area.	$\left[\left\{\begin{array}{l}\underline{\text{JUSTIFIED}}\\ \underline{\text{JUST}}\end{array}\right\}\ \text{RIGHT}\right]$

NO: 1.2.2.8

INPUT	PROCESS	OUTPUT
[BLANK WHEN ZERO]	Read entire statement, then copy to output area.	[BLANK WHEN ZERO]

NO: 1.2.2.9

INPUT	PROCESS	OUTPUT
[VALUE IS literal]	Read entire statement, then copy to output area.	[VALUE IS literal]

NO: 1.2.2.10

INPUT	PROCESS	OUTPUT
[88] condition-name {VALUE IS / VALUES ARE} literal-1 [THRU literal-2] [,literal-3 [THRU literal-4]] ...	Read entire statement, then copy to output area.	88 condition-name {VALUE IS / VALUES ARE} literal-1 [THRU literal-2] [,literal-3 [THRU literal-4]] ...

NO: 1.2.2.11

INPUT	PROCESS	OUTPUT
66 data-name-1 $\underline{\text{RENAMES}}$ data-name-2 $\left[\left\{\begin{array}{l}\underline{\text{THROUGH}}\\ \underline{\text{THRU}}\end{array}\right\}\text{data-name-3}\right]$	Read entire statement, then copy to output area.	66 data-name-1 $\underline{\text{RENAMES}}$ data-name-2 $\left[\left\{\begin{array}{l}\underline{\text{THROUGH}}\\ \underline{\text{THRU}}\end{array}\right\}\text{data-name-3}\right]$

NO: 1.2.3

INPUT	PROCESS	OUTPUT
<u>LINKAGE</u> <u>SECTION</u>.	Read entire statement, then copy to output area.	<u>LINKAGE</u> <u>SECTION</u>.

NO: 1.3

INPUT	PROCESS	OUTPUT
<u>PROCEDURE</u> <u>DIVISION</u> [<u>USING</u> data-name-1 [data-name-2] ...]	Read entire statement, then copy to output area.	<u>PROCEDURE</u> <u>DIVISION</u> [<u>USING</u> data-name-1 [data-name-2] ...]

NO: 1.3.1.1

INPUT	PROCESS	OUTPUT
<u>CALL</u> literal-1 [<u>USING</u> data-name-1 [data-name-2] ...]	Read entire statement, then copy to output area.	<u>CALL</u> literal-1 [<u>USING</u> data-name-1 [data-name-2] ...]

NO: 1.3.1.2

INPUT	PROCESS	OUTPUT
<u>CANCEL</u> $\left\{\begin{matrix}\text{id-1}\\ \text{lit-1}\end{matrix}\right\} \left\{\begin{matrix}\text{id-2}\\ \text{lit-2}\end{matrix}\right\} \ldots$	A. Read the statement until a period has been encountered. B. Since there exists a no-equivalence condition, the statement should be transferred to the output area with a warning message indicating no-equivalency. NOTE: Eliminatian of this statement may effect the program's execution.	CANCEL statement ***WARNING message

NO: 1.3.1.3

INPUT	PROCESS	OUTPUT
$\underline{\text{COPY}}\ \text{text-name}\ \left[\underline{\text{REPLACING}} \begin{Bmatrix}\text{identifier-1}\\ \text{literal-1}\\ \text{word-1}\end{Bmatrix} \underline{\text{BY}} \begin{Bmatrix}\text{identifier-2}\\ \text{literal-2}\\ \text{word-2}\end{Bmatrix}\right]\ldots$	Read entire statement, then copy to output area.	$\underline{\text{COPY}}\ \text{text-name}\ \left[\underline{\text{REPLACING}} \begin{Bmatrix}\text{identifier-1}\\ \text{literal-1}\\ \text{word-1}\end{Bmatrix} \text{BY} \begin{Bmatrix}\text{identifier-2}\\ \text{literal-2}\\ \text{word-2}\end{Bmatrix}\right]\ldots$

NO: 1.3.1.4

INPUT	PROCESS	OUTPUT
ENTER language-name [routine-name]	Read entire statement, then copy to output area.	ENTER languange-name [routine-name]

NO: 1.3.1.5

INPUT	PROCESS	OUTPUT
EXIT [PROGRAM]	Read entire statement, then copy to output area.	EXIT [PROGRAM]

NO: 1.3.1.6.1

INPUT	PROCESS	OUTPUT
<u>GO</u> <u>TO</u> [procedure-name]	Read entire statement, then copy to output area.	<u>GO</u> <u>TO</u> [procedure-name-1]

NO: 1.3.1.6.2

INPUT	PROCESS	OUTPUT
GO TO procedure-name-1 [procedure-name-2] ... DEPENDING ON identifier	Read entire statement, then copy to output area.	GO TO procedure-name-1 [procedure-name-2] ... DEPENDING ON identifier

NO: 1.3.1.7.1

INPUT

```
PERFORM   procedure-name-1
    [THRU procedure-name-2]
VARYING  { index-name-1 }
         { identifier-1 }
         { index-name-2 }
FROM     { literal-2    }
         { identifier-2 }
         { literal-3    }
  BY     { identifier-3 }
UNTIL  condition-1
[        { index-name-4 } ]
[ AFTER  { identifier-4 } ]
         { index-name-5 }
FROM     { literal-5    }
         { identifier-5 }
         { literal-6    }
  BY     { identifier-6 }
UNTIL  condition-2
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
PERFORM   procedure-name-1
    [THRU procedure-name-2]
VARYING  { index-name-1 }
         { identifier-1 }
         { index-name-2 }
FROM     { literal-2    }
         { identifier-2 }
         { literal-3    }
  BY     { identifier-3 }
UNTIL  condition-1
[        { index-name-4 } ]
[ AFTER  { identifier-4 } ]
         { index-name-5 }
FROM     { literal-5    }
         { identifier-5 }
         { literal-6    }
  BY     { identifier-6 }
UNTIL  condition-2
```

NO: 1.3.1.7.1 (con't)

INPUT	PROCESS	OUTPUT
[<u>AFTER</u> { index-name-7 / identifier-7 }		[<u>AFTER</u> { index-name-7 / identifier-7 }
<u>FROM</u> { index-name-8 / literal-8 / identifier-8 }		<u>FROM</u> { index-name-8 / literal-8 / identifier-8 }
<u>BY</u> { literal-9 / identifier-9 }		<u>BY</u> { literal-9 / identifier-9 }
<u>UNTIL</u> condition-3]]		<u>UNTIL</u> condition-3]]

NO: 1.3.1.7.2

INPUT	PROCESS	OUTPUT
PERFORM procedure-name-1 [THRU procedure-name-2] [{identifier-1 / integer-1} TIMES UNTIL condition-1]	Read entire statement, then copy to output area.	PERFORM procedure-name-1 [THRU procedure-name-2] [{identifier-1 / integer-1} TIMES UNTIL condition 1]

NO: 1.3.1.8

INPUT	PROCESS	OUTPUT
RETURN file-name RECORD [INTO identifier] AT END imperative-stmt.	Read entire statement, then copy to output area.	RETURN file-name RECORD [INTO identifier] AT END imperative-stmt.

NO: 1.3.1.9

INPUT	PROCESS	OUTPUT
START file-name INVALID KEY imperative-statement	Read entire statement, then copy to output area.	START file-name INVALID KEY imperative-statement

NO: 1.3.1.10

INPUT	PROCESS	OUTPUT
STOP RUN.	Read entire statement, then copy to output area.	STOP RUN.

NO: 1.3.1.11

INPUT

```
[illegible]
        STANDARD
[illegible]

 BEGINNING ]  [ REEL ]
 ENDING    ]  [ FILE ]
              [ UNIT ]

[illegible] PROCEDURE ON

 file-name-1 [file-name-2]...
 INPUT
 OUTPUT
 I-O
```

PROCESS

A. Read the entire statement.

B. Since there exists a no-equivalence *condition, this statement should be* transferred to the output area with a warning message indicating no-equivalency.

NOTE: Elimination of this statement may effect program's execution.

OUTPUT

USE statement

***WARNING message

NO: 1.3.2.1

INPUT	PROCESS	OUTPUT
ACCEPT identifier FROM $\left\{\begin{array}{c}\underline{\text{DATE}}\\ \underline{\text{DAY}}\\ \underline{\text{TIME}}\end{array}\right\}$	A. Read entire statement. B. Generate a SPECIAL-NAMES statement in the Data Division with a date and/or time parameter. C. Set-up a Procedure Division statement that would acquire the system's date and/or time. D. Set-up Working-Storage area to place system acquired date and/or time in appropriate format. E. Set-up a Procedure Division statement that would move the acquired date and/or time to Working-Storage area. F. Disperse generated output statements within the program in the output area.	SPECIAL-NAMES. implementor-name IS mnemonic-name ACCEPT identifier WORKING-STORAGE SECTION. 01 Date or Time, etc. MOVE identifier-1 TO identifier-2

NO: 1.3.2.2

INPUT	PROCESS	OUTPUT
CLOSE file-name-1 [{ REEL / UNIT }] [redacted] [,file-name-2 [{ REEL / UNIT }]] [WITH { NO REWIND / LOCK }] . . .	A. Read entire statement. B. If the statement contains the option: WITH { NO REWIND / LOCK } then, the remaining fields should be transferred to the output area. (NOTE: Even though this option is eliminated, the operating system or J.C.L. may have options for indicating no rewinding or locking.	CLOSE file-name-1 [{ REEL / UNIT }] [file-name-2 [{ REEL / UNIT }]] . . .

NO: 1.3.2.3

INPUT	PROCESS	OUTPUT
<u>DISPLAY</u> $\left\{\begin{array}{l}\text{identifier-1}\\ \text{literal-1}\end{array}\right\}$ $\left[\left\{\begin{array}{l}\text{identifier-2}\\ \text{literal-2}\end{array}\right\}\right]\ldots$ [redacted]	A. Read entire statement. B. If the "UPON" option is used, then it must be eliminated. (NOTE: there may exist within the operating system, means to simulate the device to display upon). This option should be moved to the output area with appropriate warning message. . Else, copy statement to output area.	UPON clause ***WARNING message <u>DISPLAY</u> $\left\{\begin{array}{l}\text{identifier-1}\\ \text{literal-1}\end{array}\right\}$ $\left[\left\{\begin{array}{l}\text{identifier-2}\\ \text{literal-2}\end{array}\right\}\right]\ldots$

NO: [illegible]

[illegible]	PR[illegible]CE[illegible]S	OUTPUT
EXHIBIT [illegible]	[illegible]	DISPLAY
[illegible]	[illegible] "EXHIBIT" [illegible] "DISPLAY".	
ident[illegible]	[illegible] the verbs "NAMED", "CHAN[illegible]", [illegible] NAMED" from the statement. [illegible] elimination [illegible] "CHANGED" and "[illegible] NAMED" might effect program's execution.	
liter[illegible]	[illegible] "NAMED" option is used, the data [illegible] must be translated [illegible] literal, [illegible] the data-name as an identifier, then add result field to the "DISPLAY" verb in the output area.	DISPLAY literal identifier
[illegible] ...	Else, add the remaining fields to the "DISPLAY" verb [illegible] the output area.	DISPLAY { identifier-1 / literal-1 } [{ identifier-2 / literal-2 }] ...
literal[illegible]		

NO: 1.3.2.5

INPUT	PROCESS	OUTPUT
OPEN { INPUT file name [[redacted]] OUTPUT file name [[redacted]] I-O file name }	A. Read entire statement. B. If the options "REVERSED" and/or "WITH NO REWIND" is used, they must be eliminated since there is a no-equivalency condition prevailing. (NOTE: There may exist in the operating system or J.C.L. options for simulating these conditions.) C. Transfer the remaining statement to the output area.	OPEN { INPUT file-name OUTPUT file-name I-O file-name } ...

NO: 1.3.2.6.1

INPUT	PROCESS	OUTPUT
READ file RECORD [INTO identifier] [INVALID KEY imperative stmt]	Read entire statement, then copy to output area.	READ file RECORD [INTO identifier] [INVALID KEY imperative stmt]

NO: 1.3.2.6.2

INPUT	PROCESS	OUTPUT
<u>READ</u> file [<u>NEXT</u>] RECORD [<u>INTO</u> identifier] [AT <u>END</u> imperative statement]	Read entire statement, then copy to output area	<u>READ</u> file [<u>NEXT</u>] RECORD [<u>INTO</u> identifier] [AT <u>END</u> imperative stmt.]

NO: 1.3.2.7

INPUT	PROCESS	OUTPUT
<u>RELEASE</u> record name [<u>FROM</u> identifier]	Read entire statement, then copy to output area.	<u>RELEASE</u> record name [<u>FROM</u> identifier]

NO: 1.3.2.8

INPUT	PROCESS	OUTPUT
RETURN file-name RECORD [INTC identifier] AT END imperative statement	Read entire statement, then copy to output area.	RETURN file-name RECORD [INTO identifier] AT END imperative statement

NO: 1.3.2.10

INPUT

```
SORT file-name-1 ON

     {ASCENDING }  KEY data-
     {DESCENDING}  name-1

     [,data-name-2] ...

[ON] {ASCENDING }  KEY data-
     {DESCENDING}  name-3

     [,data-name-4] ...] ...

{INPUT PROCEDURE IS section-
 name-1
   [{THROUGH}  section-
    {THRU   }  name-2 ]
 USING file-name-2 [,file-
 name-3] ...                }

 OUTPUT PROCEDURE IS
 section-name-3
   [{THROUGH}  section-
    {THRU   }  name-4 ]

 GIVING file-name-4
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
SORT file-name-1 ON

     {ASCENDING }  KEY data-
     {DESCENDING}  name-1

     [,data-name-2] ...

[ON] {ASCENDING }  KEY data-
     {DESCENDING}  name-3

     [,data-name-4] ...] ...

{INPUT PROCEDURE IS section-
 name-1
   [{THROUGH}  section-
    {THRU   }  name-2 ]
 USING file-name-2 [,file-
 name-3]                    }

 OUTPUT PROCEDURE IS
 section-name-3
   [{THROUGH}  section-
    {THRU   }  name-4 ]

 GIVING file-name-4
```

NO: 1.3.2.11.1

INPUT	PROCESS	OUTPUT
WRITE record-name [FROM identifier-1] [INVALID KEY imperative-statement]	Read entire statement, then copy to output area.	WRITE record-name [FROM identifier-1] [INVALID KEY imperative-statement]

NO: 1.3.2.9

INPUT	PROCESS	OUTPUT
<u>REWRITE</u> record-name [<u>FROM</u> identifier]	Read entire statement, then copy to output area.	<u>REWRITE</u> record-name [<u>FROM</u> identifier]

NO: 1.3.2.11.2

INPUT	PROCESS	OUTPUT
WRITE record name [FROM identifier-1] [{BEFORE / AFTER} ADVANCING] [{{identifier-2 / integer} LINES / mnemonic-name}] [AT {END-OF-PAGE / EOP} imperative statement]	Read entire statement, then copy to output area.	WRITE record name [FROM identifier-1] [{BEFORE / AFTER} ADVANCING] [{{identifier-2 / integer} LINES / mnemonic-name}] [AT {END-OF-PAGE / EOP} imperative statement]

AD-A107 879 SAI COMSYSTEMS CORP MCLEAN VA F/G 9/2
SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES.(U)
JUN 77 T DENIKE, A HOLLAND, T WARD, H DESAI DAHC26-76-D-1004

UNCLASSIFIED USACSC-AT-77-11 NL

2 OF 3
AD
A107879

NO: 1.3.3.1.1

INPUT	PROCESS	OUTPUT
ADD {identifier-1 } , {identifier-2} {literal-1 } {literal-2 } TO identifier-m [ROUNDED] [, identifier-n [ROUNDED]] ... [ON SIZE ERROR imperative- statement]	Read entire statement then copy to output area.	ADD identifier-1 identifier-2 literal-1 literal-2 TO identifier-m [ROUNDED] [identifier-n [ROUNDED]] ... [ON SIZE ERROR imperative- statement]

NO: 1.3.3.1.2

INPUT

```
ADD

{identifier-1} {identifier-2}
{literal-1   } {literal-2   }

  [{identifier-3}]
  [{literal-3   }] ...

GIVING identifier-m [ROUNDED]

  [identifier-n [ROUNDED]]

[ON SIZE ERROR imperative -
     statement]
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
ADD

{identifier-1} {identifier-2}
{literal-1   } {literal-2   }

  [{identifier-3}]
  [{literal-3   }] ...

GIVING identifier-m [ROUNDED]

  [identifier-n [ROUNDED]]

[ON SIZE ERROR imperative-
     statement]
```

NO: 1.3.3.2

INPUT	PROCESS	OUTPUT
COMPUTE identifier-1 [ROUNDED] [identifier-2 [ROUNDED]] ... = arithmetic-expression [ON SIZE ERROR imperative- statement]	Read entire statement, then copy to output area.	COMPUTE identifier-1 [ROUNDED] [identifier-2 [ROUNDED]] ... = arithmetic-expression [ON SIZE ERROR imperative- statement]

NO: 1.3.3.3.1

INPUT	PROCESS	OUTPUT
DIVIDE {identifier-1 / literal-1} INTO identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area.	DIVIDE {identifier-1 / literal-1} INTO identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]

NO: 1.3.3.3.2

INPUT	PROCESS	OUTPUT
<u>DIVIDE</u> {identifier-1 / literal-1} {<u>BY</u> / <u>INTO</u>} {identifier-2 / literal-2} <u>GIVING</u> identifier-3 [<u>ROUNDED</u>] [ON <u>SIZE</u> <u>ERROR</u> imperative statement]	Read entire statement, then copy to output area.	<u>DIVIDE</u> {identifier-1 / literal-1} {<u>BY</u> / <u>INTO</u>} {identifier-2 / literal-2} <u>GIVING</u> identifier-3 [<u>ROUNDED</u>] [ON <u>SIZE</u> <u>ERROR</u> imperative statement]

NO: 1.3.3.3.3

INPUT	PROCESS	OUTPUT
DIVIDE {identifier-1 / literal-1} {INTO / BY} {identifier-2 / literal-2} GIVING identifier-3 [ROUNDED] [REMAINDER ident- ifier-4] [ON SIZE ERROR imperative- statement]	Read entire statement, then copy to output area.	DIVIDE {identifier-1 / literal-1} {INTO / BY} {identifier-2 / literal-2} GIVING identifier-3 [ROUNDED] [REMAINDER iden- tifier-4] [ON SIZE ERROR imperative- statement]

NO: 1.3.3.4.1

INPUT	PROCESS	OUTPUT
MULTIPLY {identifier-1 / literal-1} BY identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area.	MULTIPLY {identifier-1 / literal-1} BY identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]

NO: 1.3.3.4.2

INPUT	PROCESS	OUTPUT

<u>MULTIPLY</u> {identifier-1 / literal-1}
<u>BY</u> {identifier-2 / literal-2}
<u>GIVING</u> identifier-3 [<u>ROUNDED</u>]
[ON <u>SIZE</u> <u>ERROR</u> imperative-statement]
	Read entire statement, then copy to output area.	<u>MULTIPLY</u> {identifier-1 / literal-1}
		<u>BY</u> {identifier-2 / literal-2}
		<u>GIVING</u> identifier-3 [<u>ROUNDED</u>]
		[ON <u>SIZE</u> <u>ERROR</u> imperative-statement]

NO: 1.3.3.5.1

INPUT	PROCESS	OUTPUT
SUBTRACT {identifier-1 | literal-1} [identifier-2 | literal-2] ... FROM {identifier-m | literal-m} GIVING identifier-n [ROUNDED] [ON SIZE ERROR imperative statement]	Read entire statement, then copy to output area.	SUBTRACT {identifier-1 | literal-1} [identifier-2 | literal-2] ... FROM {identifier-m | literal-m} GIVING identifier-n [ROUNDED] [ON SIZE ERROR imperative statement]

NO: 1.3.3.5.2

INPUT

```
SUBTRACT   { identifier-1 }
           { literal-1    }

     [ identifier-2 ]
     [ literal-2    ]  ...

FROM identifier-m [ROUNDED]

  [ identifier-n [ROUNDED] ] ...

  [ON SIZE ERROR imperative-
        statement ]
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
SUBTRACT   { identifier-1 }
           { literal-1    }

     [ identifier-2 ]
     [ literal-2    ]  ...

FROM identifier-m [ROUNDED]

  [ identifier-n [ROUNDED] ] ...

  [ON SIZE ERROR imperative-
        statement ]
```

NO: [illegible]

INPUT

```
EXAMINE id TALLYING

  { UNTIL FIRST }
  { ALL         }  lit-1
  { LEADING     }

  REPLACING BY lit-2


EXAMINE id REPLACING

  { ALL         }
  { LEADING     }  lit-1 BY
  { FIRST       }  lit-2
  { UNTIL FIRST }
```

PROCESS

A. Read entire statement.

B. Transform the word EXAMINE to the word INSPECT.

C. Carry over "id TALLYING" or "id REPLACING" fields and add to INSPECT verb.

D. If the "UNTIL FIRST" option has been used in either format, an "IF" condition must be set-up to check, within the Procedure Division.

E. If other options are used, add to the INSPECT format and transfer entire statement(s) to the output area.

OUTPUT

```
INSPECT

INSPECT id TALLYING
      (or)
INSPECT id REPLACING

IF first occurance THEN
imperative statement

INSPECT id TALLYING
 { ALL     }     lit-1
 { LEADING }
  REPLACING BY lit-2
            (or)
INSPECT id REPLACING
  ALL
  LEADING   lit-1 BY lit-2
  FIRST
```

NO: 1.3.4.2.1

INPUT	PROCESS	OUTPUT
MOVE $\left\{\begin{array}{l}\text{identifier-1}\\\text{literal}\end{array}\right\}$ TO identifier-2 [identifier-3] ...	Read entire statement, then copy to output area.	MOVE $\left\{\begin{array}{l}\text{identifier-1}\\\text{literal}\end{array}\right\}$ TO identifier-2 [identifier-3] ...

NO: 1.3.4.2.2

INPUT	PROCESS	OUTPUT
$\underline{\text{MOVE}} \left\{ \begin{array}{l} \underline{\text{CORRESPONDING}} \\ \underline{\text{CORR}} \end{array} \right\}$ identifier-1 $\underline{\text{TO}}$ identifier-2	Read entire statement, then copy to output area.	$\underline{\text{MOVE}} \left\{ \begin{array}{l} \underline{\text{CORRESPONDING}} \\ \underline{\text{CORR}} \end{array} \right\}$ identifier-1 $\underline{\text{TO}}$ identifier-2

NO: 1.3.4.3.1

INPUT

```
SEARCH  identifier-1

 [VARYING {identifier-2 }]
 [        {index-name-1 }]

 [AT END imperative-stmt.]

WHEN condition-1

    {imperative-stmt}
    {NEXT SENTENCE  }

 [WHEN condition-2

    {imperative-stmt}
    {NEXT SENTENCE  }] ...
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
SEARCH  identifier-1

[VARYING {identifier-2 }]
[        {index-name-1 }]

 [AT END imperative-stmt.]

WHEN condition-1

    {imperative-stmt}
    {NEXT SENTENCE  }

  [WHEN condition-2

    {imperative-stmt}
    {NEXT SENTENCE  }]
```

NO: 1.3.4.3.

INPUT	PROCESS	OUTPUT
SEARCH ALL identifier-1 [AT END imperative stmt-1] WHEN { data-name-1 { IS EQUAL TO / IS = } { identifier-2 / literal-1 / arithmetic-expression-1 } / condition-name-1 } [AND { data-name-2 { IS EQUAL TO / IS = } { identifier-3 / literal-2 / arithmetic-expression-2 } / condition-name-2 }] ... { imperative-stmt-2 / NEXT SENTENCE }	Read entire statement, then copy to output area.	SEARCH ALL identifier-1 [AT END imperative stmt-1] WHEN { data-name-1 { IS EQUAL TO / IS = } { identifier-2 / literal-1 / arithmetic-expression-1 } / condition-name-1 } [AND { data-name-2 { IS EQUAL TO / IS = } { identifier-3 / literal-2 / arithmetic-expression-2 } / condition-name-2 }] ... { imperative-stmt-2 / NEXT SENTENCE }

NO: 1.3.4.4.1

INPUT	PROCESS	OUTPUT
<u>SET</u> { identifier-1 / index-name-1 } [, identifier-2 / , index-name-2] ... <u>TO</u> { identifier-3 / index-name-3 / integer-1 }	Read entire statement, then copy to output area.	<u>SET</u> { identifier-1 / index-name-1 } [, identifier-2 / , index-name-2] ... <u>TO</u> { identifier-3 / index-name-3 / integer-3 }

INPUT	PROCESS	OUTPUT
SET	Send entire statement, then copy to output area.	SET
index-name-4 [, index-name-5] ...		index-name-4 [, index-name-5] ...
UP BY {identifier-4}		{UP BY} {identifier-4}
DOWN BY {integer-2}		{DOWN BY} {integer-2}

NO: 1.3.5.1

INPUT	PROCESS	OUTPUT
$\underline{\text{IF}}$ condition $\left\{\begin{array}{l}\text{statement-1}\\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}}\end{array}\right\}$ $\underline{\text{ELSE}}\ \left\{\begin{array}{l}\text{statement-2}\\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}}\end{array}\right\}$	Read entire statement, then copy to output area.	$\underline{\text{IF}}$ condition $\left\{\begin{array}{l}\text{statement-1}\\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}}\end{array}\right\}$ $\underline{\text{ELSE}}\ \left\{\begin{array}{l}\text{statement-2}\\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}}\end{array}\right\}$

NO: 1.3.3.2

INPUT

```
ON int-1  [AND EVERY int-2]
    [UNTIL int-2]
    {imp-stmt     }
    {NEXT SENTENCE}
   {ELSE     }
   {OTHERWISE}
    {statement    }
    {NEXT SENTENCE}
```

PROCESS

A. Read entire statement.

B. Generate Working Storage statements.

C. Set-up initialization statement for counter-field in Procedure Division.

D. Set-up an adding statement to the counter-field for incrementing.

E. Set-up "IF" statement for checking in re-menting process in previous step.

OUTPUT

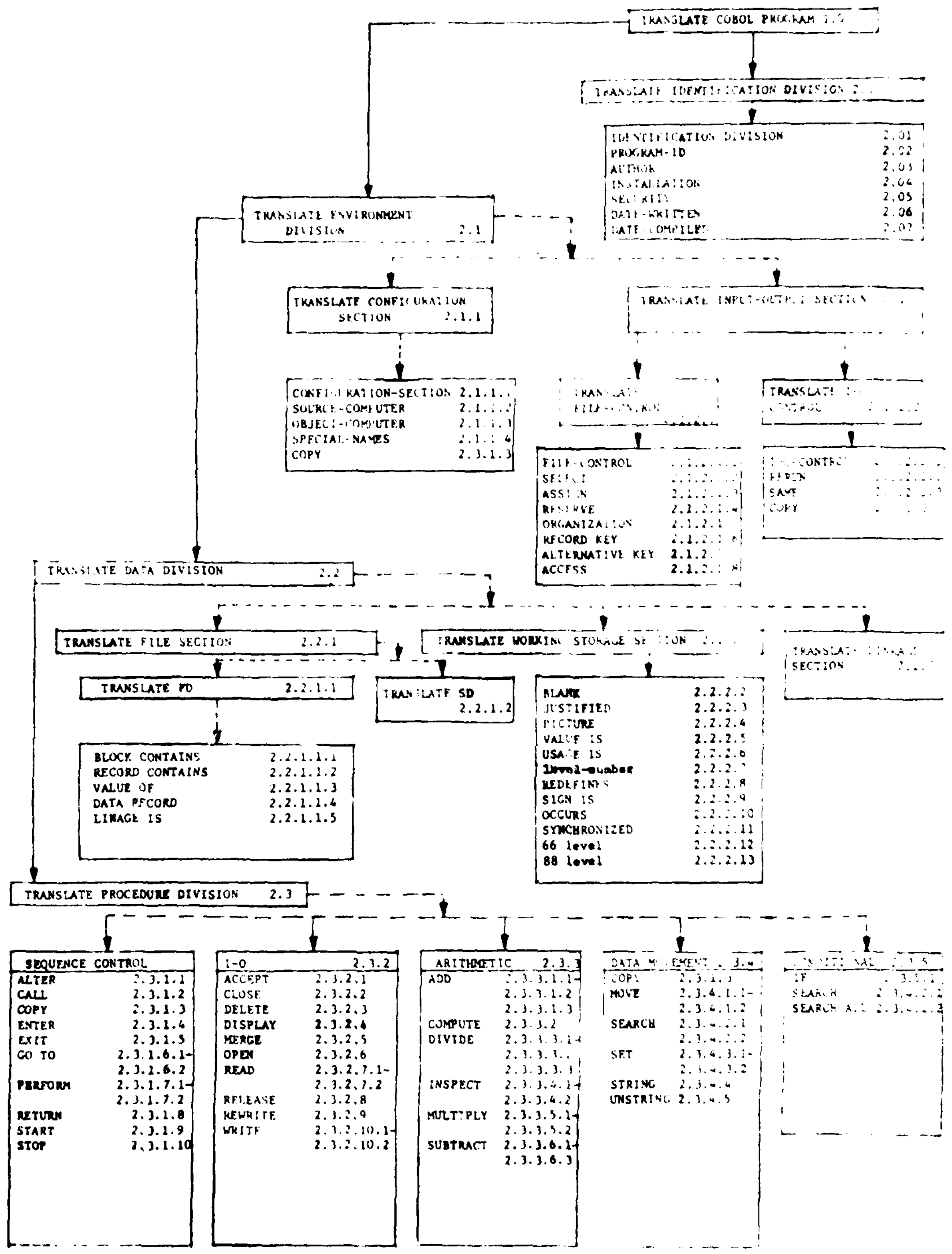
```
77 argument-field PIC 9 (8).
77 counter-field  PIC 9 (8).
77 ONE       PIC 9   VALUE 1.

MOVE ZEROS TO counter-field.

ADD ONE TO counter-field.

IF counter-field = (int-1 +
(int-2 * argument field)
AND counter-field      int-3
           imp-stmt
  THEN
           NEXT SENTENCE

           imp-stmt
  ELSE
           NEXT SENTENCE
```

APPENDIX B

PORTABLE STANDARD COBOL (PSC)

TO

PDP 11 COBOL


TRANSLATE COBOL PROGRAM
TRANSLATE IDENTIFICATION DIVISION
IDENTIFICATION DIVISION 2.01
PROGRAM-ID 2.02
AUTHOR 2.03
INSTALLATION 2.04
SECURITY 2.05
DATE-WRITTEN 2.06
DATE-COMPILED 2.07
TRANSLATE ENVIRONMENT DIVISION 2.1
TRANSLATE CONFIGURATION SECTION 2.1.1
CONFIGURATION-SECTION 2.1.1.1
SOURCE-COMPUTER 2.1.1.2
OBJECT-COMPUTER 2.1.1.3
SPECIAL-NAMES 2.1.1.4
COPY 2.3.1.3
TRANSLATE INPUT-OUTPUT SECTION
TRANSLATE FILE-CONTROL
FILE-CONTROL
SELECT
ASSIGN
RESERVE
ORGANIZATION 2.1.2.1
RECORD KEY
ALTERNATIVE KEY 2.1.2.
ACCESS
TRANSLATE I-O CONTROL
I-O-CONTROL
RERUN
SAME
COPY
TRANSLATE DATA DIVISION 2.2
TRANSLATE FILE SECTION 2.2.1
TRANSLATE FD 2.2.1.1
BLOCK CONTAINS 2.2.1.1.1
RECORD CONTAINS 2.2.1.1.2
VALUE OF 2.2.1.1.3
DATA RECORD 2.2.1.1.4
LINAGE IS 2.2.1.1.5
TRANSLATE SD 2.2.1.2
TRANSLATE WORKING STORAGE SECTION
BLANK 2.2.2.2
JUSTIFIED 2.2.2.3
PICTURE 2.2.2.4
VALUE IS 2.2.2.5
USAGE IS 2.2.2.6
level-number 2.2.2.7
REDEFINES 2.2.2.8
SIGN IS 2.2.2.9
OCCURS 2.2.2.10
SYNCHRONIZED 2.2.2.11
66 level 2.2.2.12
88 level 2.2.2.13
TRANSLATE LINKAGE SECTION
TRANSLATE PROCEDURE DIVISION 2.3
SEQUENCE CONTROL
ALTER 2.3.1.1
CALL 2.3.1.2
COPY 2.3.1.3
ENTER 2.3.1.4
EXIT 2.3.1.5
GO TO 2.3.1.6.1-2.3.1.6.2
PERFORM 2.3.1.7.1-2.3.1.7.2
RETURN 2.3.1.8
START 2.3.1.9
STOP 2.3.1.10
I-O 2.3.2
ACCEPT 2.3.2.1
CLOSE 2.3.2.2
DELETE 2.3.2.3
DISPLAY 2.3.2.4
MERGE 2.3.2.5
OPEN 2.3.2.6
READ 2.3.2.7.1-2.3.2.7.2
RELEASE 2.3.2.8
REWRITE 2.3.2.9
WRITE 2.3.2.10.1-2.3.2.10.2
ARITHMETIC 2.3.3
ADD 2.3.3.1.1-2.3.3.1.2 2.3.3.1.3
COMPUTE 2.3.3.2
DIVIDE 2.3.3.3.1-2.3.3.3.3
INSPECT 2.3.3.4.1-2.3.3.4.2
MULTIPLY 2.3.3.5.1-2.3.3.5.2
SUBTRACT 2.3.3.6.1-2.3.3.6.3
DATA MOVEMENT 2.3.4
COPY
MOVE 2.3.4.1.1-2.3.4.1.2
SEARCH 2.3.4.2.1 2.3.4.2.2
SET 2.3.4.3.1-2.3.4.3.2
STRING 2.3.4.4
UNSTRING 2.3.4.5
CONDITIONAL 2.3.5
IF
SEARCH
SEARCH ALL

NO: 2.0.1

INPUT	PROCESS	**OUTPUT**
<u>IDENTIFICATION DIVISION</u>.	Read entire statement, then copy to output area.	<u>IDENTIFICATION DIVISION.</u>

NO: 2.0.2

INPUT	PROCESS	OUTPUT
PROGRAM-ID. program-name.	Read entire statement, then copy to output area.	PROGRAM-ID. program-name.

NO: 2.0.3

INPUT	PROCESS	OUTPUT
[AUTHOR.] [[comment-entry] ...]	Read entire statement, then copy to output area.	[AUTHOR.] [comment-entr]

NO: 2.0.4

INPUT	PROCESS	OUTPUT
INSTALLATION. [Comment-entry] . . .	Read entire statement, then copy to output area.	INSTALLATION. [Comment-entry] . . .

NO: 2.0.5

INPUT	PROCESS	OUTPUT
[<u>SECURITY</u>.] [[comment-entry] ...]	Read entire statement, then copy to output area.	[<u>SECURITY</u>.] [[comment-entry] ...]

NO: 2.0.6

INPUT	PROCESS	OUTPUT
[DATE-WRITTEN.] [comment-entry] ...	Read entire statement, then copy to output area.	[DATE-WRITTEN.] [comment-entry] ...

NO: 2.0.7

INPUT	PROCESS	OUTPUT
[DATE-COMPILED.] [[comment-entry] ...]	Read entire statement, then copy to output area.	[DATE-COMPILED.] [[comment-entry] ...]

NO: 2.1

INPUT	PROCESS	OUTPUT
ENVIRONMENT DIVISION.	Read entire statement, then copy to output area.	ENVIRONMENT DIVISION.

NO: 2.1.1.1

INPUT	PROCESS	OUTPUT
CONFIGURATION SECTION.	Read entire statement, then copy to output area	CONFIGURATION SECTION.

NO: 2.1.1.2

INPUT	PROCESS	OUTPUT
<u>SOURCE-COMPUTER.</u> computer-name	Read entire statement, then copy to output area.	<u>SOURCE-COMPUTER.</u> computer-name

NO: 2.1.1.3

INPUT	PROCESS	OUTPUT
<u>OBJECT-COMPUTER</u>. computer-name [<u>MEMORY</u> <u>SIZE</u> integer { <u>WORDS</u> / <u>CHARACTERS</u> / <u>MODULES</u> }] [redacted] [redacted]	A. Read entire statement. B. If "Segment-Limit" clause exists, move clause to output area with warning message (NOTE: elimination of this clause will not effect program execution.) C. Else, copy statement (or remaining statement with "SEGMENT" clause eliminated) to output area.	<u>OBJECT-COMPUTER.</u> computer-name [<u>MEMORY</u> <u>SIZE</u> integer { <u>WORDS</u> / <u>CHARACTERS</u> / <u>MODULES</u> }] SEGMENT clause ***warning message

INPUT

```
SPECIAL-NAMES.
    implementor-name

    IS mnemonic-name

     ON STATUS IS cond-1
        [OFF STATUS IS cond-2]
     OFF STATUS IS cond-2
        [ON STATUS IS cond-1]
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
SPECIAL-NAMES.
   [implementor-name IS
       mnemonic-name]
```

[redacted]

```
  ON STATUS IS cond-1
     [OFF STATUS IS cond-2]
  OFF STATUS IS cond-2
     [ON STATUS IS cond-2]      ...
```

NO: 2.1.2

INPUT	PROCESS	OUTPUT
<u>INPUT-OUTPUT</u> <u>SECTION.</u>	Read entire statement, then copy to output area.	<u>INPUT-OUTPUT</u> <u>SECTION.</u>

NO: 2.1.2.1.1

INPUT	PROCESS	OUTPUT
<u>FILE-CONTROL.</u>	Read entire statement, then copy to output area.	<u>FILE-CONTROL.</u>

NO: 2.1.2.[illegible]

INPUT	PROCESS	OUTPUT
<u>SELECT</u> file-name.	Read entire statement, then copy to output area.	<u>SELECT</u> file-name.

INPUT	PROCESS	OUTPUT
<u>ASSIGN</u> TO implementor-name-1 [, implementor-name-2] ...	Read entire statement, then copy to output area.	<u>ASSIGN</u> TO implementor-name-1 [, implementor-name-2] ...

NO: 2.1.2.1.4

INPUT	PROCESS	OUTPUT
[<u>RESERVE</u> integer [AREA / AREAS]]	Read entire statement, then copy to output area.	[<u>RESERVE</u> integer [AREA / AREAS]]

INPUT	PROCESS	OUTPUT
<u>ORGANIZATION</u> IS { <u>RELATIVE</u> / <u>SEQUENTIAL</u> / [redacted] }	A. Read entire statement B. If organization is "INDEXED", move statement and warning message to output area. C. If organization is "RELATIVE" or "SEQUENTIAL", move statement to output area, and copy.	<u>ORGANIZATION</u> IS { <u>RELATIVE</u> / <u>SEQUENTIAL</u> } ***Warning message. Indexed option is not allowed.

NO: 2.1.2.1.6

INPUT	PROCESS	OUTPUT
RECORD KEY IS data-name	A. Read entire statement B. Since this caluse is a no-equivalent situation, it should be moved to the output area with appropriate warning message. NOTE: There may exist within the operating system, the means to simulate this process, since elimination of this clause may effect execution.	RECORD statement ***WARNING message

INPUT

```
ALTERNATE RECORD KEY
   IS data-name
   [WITH DUPLICATES]
```

PROCESS

A. Read entire statement

B. Since there is no equivalent statement within the output language, this "ALTERNATE" statement should be moved to the output area with appropriate warning message.

NOTE: There may exist means within the operating system to simulate this statement as function, because elimination of statement may effect program's execution.

OUTPUT

```
ALTERNATE RECORD KEY
   IS data-name
   [WITH DUPLICATES]

*** WARNING message
```

INPUT	PROCESS	OUTPUT
[ACCESS, MODE IS] { SEQUENTIAL [RELATIVE KEY IS data-name-1] {RANDOM/DYNAMIC} RELATIVE KEY IS data-name-2 }	Read entire statement, then copy to output area	[ACCESS MODE IS] { SEQUENTIAL [RELATIVE KEY IS data-name-1] {RANDOM/DYNAMIC} RELATIVE KEY IS data-name-2 }

NO: 2.1.2.2.1

INPUT	PROCESS	OUTPUT
I-O-CONTROL.	Read entire statement, then copy to output area.	I-O-CONTROL.

NO: 2.1.2.2.2

INPUT

```
RERUN  ON

  [ file-name-1          ]
  [ implementor-name     ]

  [ { END OF { REEL }              } ]
  [ {        { UNIT }              } ]
  [ {                              } ]
  [ {        OF                    } ]
  [ {        file-name-2           } ]
  [ {                              } ]
  [ { integer-1 RECORDS            } ]
  [                                  ] ...
  [ integer-2 CLOCK UNITS            ]
  [                                  ]
  [ condition-name                   ]
```

PROCESS

A. Read entire statement

B. Since there isn't an equivalent statement within the output language, this statement should be eliminated and placed in the output area with appropriate error message.

NOTE: There may exist in the operating system the means for simulating this concept, because elimination of this statement may effect simulation.

OUTPUT

RERUN statement

***WARNING message

NO: 2.1.2.2.3

INPUT	PROCESS	OUTPUT
[SAME AREA FOR] file-name-1 [file-name-2] ...	Read Entire statement then copy to output area.	[SAME AREA FOR] file-name-1 [, file-name-2] ...

NO: 2.2

INPUT	PROCESS	OUTPUT
DATA DIVISION.	Read entire statement, then copy to output area	DATA DIVISION.

NO: 2.2.1

INPUT	PROCESS	OUTPUT
FILE SECTION.	Read entire statement, then copy to output area	FILE SECTION.

NO: 2.2.1.1

INPUT	PROCESS	OUTPUT
FD file-description	Read entire statement and copy to output area	FD file-description

NO: 2.2.1.1.1

INPUT	PROCESS	OUTPUT
[<u>BLOCK</u> <u>CONTAINS</u> integer-2 [RECORDS / <u>CHARACTERS</u>]]	Read entire statement, then copy to output area	[<u>BLOCK</u> <u>CONTAINS</u> integer-2 {<u>RECORDS</u> / CHARACTERS}]

NO: 2.2.1.1.2

INPUT	PROCESS	OUTPUT
<u>RECORD</u> <u>CONTAINS</u> [integer-3 <u>TO</u>] integer-4 CHARACTERS	Read entire statement, then copy to output area	<u>RECORD</u> <u>CONTAINS</u> [integer-3 <u>TO</u>] integer-4 CHARACTERS

NO: 2.2.1.1.3

INPUT	PROCESS	OUTPUT
VALUE OF implementor-name-1 IS literal-1 [redacted] [redacted] . . .	A. *Read entire statement* B. Move "VALUE OF" to output area and translate implementor-name-1 to "ID" within output area. C. Move "IS literal-1" to output area. D. If the optional implementor phrase has been used, it should be moved to the output area with a warning message (NOTE: This phrase's elimination may effect program execution).	VALUE OF ID VALUE OF ID IS literal-1 implementor option phrase ***WARNING message

NO: 2.2.1.1.4

INPUT	PROCESS	OUTPUT
$\left[\underline{\text{DATA}}\left\{\begin{array}{l}\underline{\text{RECORD}}\text{ IS}\\\underline{\text{RECORDS}}\text{ ARE}\end{array}\right\}\right]$ data-name-3 $\left[\text{, data-name-4}\right]\ldots$	Read entire statement, then copy to output area	$\left[\underline{\text{DATA}}\left\{\begin{array}{l}\underline{\text{RECORD}}\text{ IS}\\\underline{\text{RECORDS}}\text{ ARE}\end{array}\right\}\right]$ data-name-3 $\left[\text{, data-name-4}\right]\ldots$

NO:

INPUT	PROCESS	OUTPUT
[<u>LINAGE</u> IS {data-name-5 / integer-5}] LINES [WITH <u>FOOTING</u> AT {data-name-6 / integer-6}] [, LINES AT <u>TOP</u> {data-name-7 / integer-7}] [, LINES AT <u>BOTTOM</u> {data-name-8 / integer-8}]	*Read entire statement, then copy to output area*	[<u>LINAGE</u> IS {data-name-5 / integer-5}] LINES [WITH <u>FOOTING</u> AT {data-name-6 / integer-6}] [, LINES AT <u>TOP</u> {data-name-7 / integer-7}] [, LINES AT <u>BOTTOM</u> {data-name-8 / integer-8}]

NO: 2.2.1.2

INPUT	PROCESS	OUTPUT
SD file-name	A. Read entire statement, with succeeding entries.	
	B. Since there exists no equivalent statement within the output language, this "SD" statement, along with succeeding entries, should be moved to the output area with warning messages.	SD statement ***WARNING message level descriptions ***WARNING messages
	NOTE: The operating system may contain the means to simulate this process, since eliminating these statements may effect program execution.	

NO: 2.2.2

INPUT	PROCESS	OUTPUT
WORKING-STORAGE SECTION.	Read entire statement, then copy to output area	WORKING-STORAGE SECTION.

NO: 2.2.2.2

INPUT	PROCESS	OUTPUT
BLANK WHEN ZERO	Read entire statement, then copy to output area	[BLANK WHEN ZERO]

NO: 2.2.2.3

INPUT	PROCESS	OUTPUT
[<u>JUSTIFIED</u> / <u>JUST</u>] RIGHT	Read entire statement, then copy to output area	[{ <u>JUSTIFIED</u> / <u>JUST</u> }] RIGHT

NO: 2.2.2.4

INPUT	PROCESS	OUTPUT
$\left\{\begin{array}{l}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{array}\right\}$ IS character string	Read entire statement, then copy to output area	$\left\{\begin{array}{l}\underline{\text{PICTURE}}\\ \underline{\text{PIC}}\end{array}\right\}$ IS character string

NO: 2.2.2.5

INPUT	PROCESS	OUTPUT
VALUE IS literal	Read entire statement, then copy to output area	VALUE IS literal

NO: 2.2.2.6

INPUT	PROCESS	OUTPUT
[[USAGE IS] { COMPUTATIONAL COMP DISPLAY INDEX }]	Read entire statement, then copy to output area	[[USAGE IS] { COMPUTATIONAL COMP DISPLAY INDEX }]

NO: 2.2.2.7

INPUT	PROCESS	OUTPUT
level-number $\left\{\begin{array}{l}\text{data-name-1}\\ \underline{\text{FILLER}}\end{array}\right\}$	Read entire statement, then copy to output area	level-number $\left\{\begin{array}{l}\text{data-name-1}\\ \underline{\text{FILLER}}\end{array}\right\}$

NO: 2.2.2.8

INPUT	PROCESS	OUTPUT
[<u>REDEFINES</u> data-name-2]	Read entire statement, then copy to output area	[<u>REDEFINES</u> data-name-2]

NO: 2.2.2.9

INPUT	PROCESS	OUTPUT
[SIGN IS {LEADING / TRAILING}] [SEPARATE CHARACTER]	*Read entire statement, then copy to output area*	[SIGN IS {LEADING / TRAILING}] [SEPARATE CHARACTER]

NO: 2.2.2.10

INPUT	PROCESS	OUTPUT
[OCCURS {integer-1 TO integer-2 TIMES {integer-2 TIMES DEPENDING ON data-name-3] [{ASCENDING DESCENDING} KEY IS data-name-4 [data-name-5] ...] ... [INDEXED BY index-name-1 [index-name ...]]	Read entire statement, then copy to output area	[OCCURS integer-1 TO integer-2 TIMES integer-2 TIMES DEPENDING ON data-name-3] [{ASCENDING DESCENDING} KEY IS data-name-4 [data-name-5] ...] ... [INDEXED BY index-name-1 [index-name ...]]

INPUT	PROCESS	OUTPUT
{ SYNCHRONIZED / SYNC } [LEFT / RIGHT]	Read entire statement, then copy to the output area	{ SYNCHRONIZED / SYNC } [LEFT / RIGHT]

NO: 2.2.2.12

INPUT	PROCESS	OUTPUT
66 data-name-1 $\underline{\text{RENAMES}}$ data-name-2 $\left[\begin{Bmatrix}\underline{\text{THROUGH}}\\ \underline{\text{THRU}}\end{Bmatrix}\text{ data-name-3}\right]$	Read entire statement, then copy to output area	66 data-name-1 $\underline{\text{RENAMES}}$ data-name-2 $\left[\begin{Bmatrix}\underline{\text{THROUGH}}\\ \underline{\text{THRU}}\end{Bmatrix}\text{ data-name-3}\right]$

INPUT

```
88 condition-name

   VALUE IS
   VALUES ARE     literal-1

THROUGH
THRU              literal-2

literal-3         THROUGH
                  THRU

   literal-4
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
88 condition-name

   VALUE IS
   VALUES ARE     literal-1

THROUGH
THRU              literal-2

literal-3         THROUGH
                  THRU

   literal-4
```

INPUT	PROCESS	OUTPUT
<u>LINKAGE</u> <u>SECTION</u>	A. Read **entire** statement, with succeeding entries: B. **Since there exists** no equivalent statement **within the output** language, this "LINKAGE" **statement, along** with succeeding **statements, should be** moved to the output area with warning messages. NOTE: The **operating** system may contain the means for **simulating** this concept, since eliminating these **statements** may effect execution.	LINKAGE statement with other entires ***WARNING message

NO: 2.3.1

INPUT	PROCESS	OUTPUT
PROCEDURE DIVISION [illegible] [illegible] . . .	A. Read entire statement. B. Move "PROCEDURE DIVISION" to the output area. C. If the "USING data-name-1" option is used, then it should be moved to the output area with appropriate warning message. NOTE: Elimination of this clause may effect execution, but, the operating system may contain means to simulate the "USING" process.	PROCEDURE DIVISION USING clause ***WARNING message

NO: 2.3.1.1

INPUT	PROCESS	OUTPUT
ALTER procedure-name-1 TO [PROCEED TO] procedure-name-2 [procedure-name-3 TO [PROCEED TO] procedure-name-4] . . .	Read entire statement, then copy to output area	ALTER procedure-name-1 TO [PROCEED TO] procedure-name-2 [procedure-name-3 TO [PROCEED TO] procedure-name-4] . . .

NO: 2.3.1.2

INPUT	PROCESS	OUTPUT
CALL literal-1 [USING data-name-1 [, data-name-2] ...]	A. Read entire statement B. Since there is no equivalent, this statement should be eliminated, and sent to the output area with appropriate warning message. NOTE: Elimination of this statement may effect program execution. Within particular operating systems, there exists means to perform calling or linking modules.	CALL statement *** WARNING message

NO: 2.3.1.3

INPUT	PROCESS	OUTPUT
COPY text-name [REPLACING { [redacted] literal-1 word-1 } BY { identifier-2 literal-2 word-2 } } ...]	A. Read entire statement. B. If statement contains "identifier" option, then move statement to output area with warning message. (NOTE: Eliminating statement may effect execution of program) C. Else, copy statement to output area.	COPY text-name REPLACING identifier-1 BY identifier-2 ***WARNING message COPY text-name [REPLACING { { literal-1 word-1 } BY { literal-2 word-2 } } ...]

INPUT	PROCESS	OUTPUT
ENTER language-name routine-name	A. Read entire statement B. Since there is no equivalent verb for the output source, move the statement to the output area with appropriate error message. NOTE: *Elimination of this statement may effect execution, but there may exist means within the operating system to similate this "ENTER" process.*	ENTER statement *** WARNING message

NO: 2.3.1.5

INPUT	PROCESS	OUTPUT
EXIT	Read entire statement, then copy to output area.	EXIT

NO: 2.3.1.6.1

INPUT	PROCESS	OUTPUT
GO TO [procedure-name-1]	Read entire statement, then copy to output area.	GO TO [procedure-name-1]

NO: 2.3.1.6.2

INPUT	PROCESS	OUTPUT
GO TO procedure-name-1 [,procedure-name-2] ... ,procedure-name-n DEPENDING ON identifier	Read entire statement, then copy to output area.	GO TO procedure-name-1 [,procedure-name-2] ... ,procedure-name-n DEPENDING ON identifier

INPUT	PROCESS	OUTPUT
PERFORM procedure-name-1 $\left[\underline{\text{THRU}}\ \text{procedure-name-2}\right]$ $\left[\begin{array}{l}\left\{\begin{array}{l}\text{identifier-1}\\ \text{integer-1}\end{array}\right\}\ \underline{\text{TIMES}}\\ \underline{\text{UNTIL}}\ \text{condition-1}\end{array}\right]$	Read entire statement, then copy to output area	$\underline{\text{PERFORM}}$ procedure-name-1 $\left[\underline{\text{THRU}}\ \text{procedure-name-2}\right]$ $\left[\begin{array}{l}\left\{\begin{array}{l}\text{identifier-1}\\ \text{integer-1}\end{array}\right\}\ \underline{\text{TIMES}}\\ \underline{\text{UNTIL}}\ \text{condition-1}\end{array}\right]$

NO: 2.3.1.8

INPUT	PROCESS	OUTPUT
RETURN file-name RECORD [INTO identifier] AT END imperative- statement	A. Read entire statement B. There exists no equivalent statement in the output language. This statement should be eliminated, and moved to the output area with appropriate warning message. NOTE: There may exist means within the JCL or operating system, which simulates this RETURN function. Eliminating this statement may effect execution.	RETURN statement ***WARNING message

NO: 2.3.1.9

INPUT	PROCESS	OUTPUT
START file-name [KEY { IS EQUAL TO IS = IS GREATER THAN IS IS NOT LESS THAN IS NOT } data-name]	A. Read entire statement and copy as is	START file-name [KEY { IS EQUAL TO IS = IS GREATER THAN IS IS NOT LESS THAN IS NOT } data-name]

NO: 2.3.1.10

INPUT	PROCESS	OUTPUT
<u>STOP</u> $\left\{\begin{array}{l}\underline{\text{RUN}}\\ \text{literal}\end{array}\right\}$	A. Read entire statement and copy as is	<u>STOP</u> $\left\{\begin{array}{l}\underline{\text{RUN}}\\ \text{literal}\end{array}\right\}$

NO: 2.3.2.1

INPUT	PROCESS	OUTPUT
ACCEPT identifier [FROM mnemonic-name]	Read entire statement then copy to output area	ACCEPT identifier [FROM mnemonic-name]

NO: 2.3.2.2

INPUT	PROCESS	OUTPUT
<u>CLOSE</u> file-name-1 [,file-name-2]	Read entire statement, then copy to output area	<u>CLOSE</u> file-name-1 [,file-name-2]

NO: 2.3.2.3

	PROCESS	OUTPUT
RE KEY perative-statement	Read entire statement, then copy to output area	DELETE file-name RECORD [INVALID KEY imperative-statement]

NO: 2.3.2.4

INPUT	PROCESS	OUTPUT
$\underline{\text{DISPLAY}} \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[\begin{array}{l} \text{, identifier-2} \\ \text{, literal-2} \end{array} \right] \ldots$	Read entire statement, then copy to output area	$\underline{\text{DISPLAY}} \left\{ \begin{array}{l} \text{identifier-1} \\ \text{literal-1} \end{array} \right\} \left[\begin{array}{l} \text{, identifier-2} \\ \text{, literal-2} \end{array} \right] \ldots$

INPUT	PROCESS	OUTPUT
MERGE file-name-1 ON {ASCENDING / DESCENDING} KEY data-name-1 [, data-name-2] . . . [ON {ASCENDING / DESCENDING} KEY data-name-3 [, data-name-4] . . .] . . . USING file-name-2, file-name-3 [, file-name-4] . . . (CONTINUED ON NEXT PAGE)	A. Read entire statement B. Since there is no equivalent statement, then this statement should be moved to the output area with a warning message. NOTE: To simulate this merge function, a series of statements for reading, checking and writing may be set-up for the multiple files, but, efficiency may be lost.	MERGE statement ***WARNING message

INPUT	PROCESS	OUTPUT
OUTPUT PROCEDURE IS		READ file-name-1
section-name-1		READ file-name-2
GIVING file-name-5		IF condition-1 THEN
[{THROUGH / THRU} section-name-2]		imperative-statement.
		(where the ASCENDING or
		DESCENDING condition
		is simulated and/or
		checking
		WRITE file-name-3

NO: 2.3.2.6

INPUT	PROCESS	OUTPUT
OPEN { INPUT file-name-1 OUTPUT file-name-3 I-O file-name-5 } [,file-name-2] . . . [,file-name-4] . . . [,file-name-6] . . .	Read entire statement, then copy to output area	OPEN { INPUT file-name-1 OUTPUT file-name-3 I-O file-name-5 } [,file-name-2] . . . [,file-name-4] . . . [,file-name-6] . . .

NO: 2.3.2.7.1

INPUT	PROCESS	OUTPUT
READ file-name [NEXT] RECORD [INTO identifier] [AT END imperative- statement]	Read entire statement, then copy to output area	READ file-name [NEXT] RECORD [INTO identifier] [AT END imperative- statement]

NO: 2.3.2.7.2

INPUT	PROCESS	OUTPUT
READ file-name RECORD	A. Read entire statement	
[INTO identifier]	B. If "KEY IS" option is used, then it must be eliminated from the output language, by moving clause to output area with appropriate warning message. (NOTE: Elimination of this clause may effect execution.)	KEY clause
[redacted]	C. Else copy statement to output area	***WARNING statement
[INVALID KEY		READ file-name RECORD
imperative-statement]		[INTO identifier]
		[INVALID KEY
		imperative-statement]

NO: 2.3.2.8

INPUT	PROCESS	OUTPUT
RELEASE record-name [FROM identifier]	A. Read entire statement B. There isn't an equivalent statement in the output language. This statement should be eliminated and moved tc the output area with appropriate warning message. NOTE: There may exist means within the operating system, which may simulate this RELEASE function. Eliminating this statement may effect execution.	RELEASE statement ***WARNING message

NO: 2.3.2.9

INPUT	PROCESS	OUTPUT
REWRITE record-name [FROM identifier] [INVALID KEY imperative-statement]	Read entire statement, then copy to output area	REWRITE record-name [FROM identifier] [INVALID KEY imperative-statement]

NO: 2.3.2.10.1

INPUT

```
WRITE record-name
   [FROM identifier-1]
  [{BEFORE}  ADVANCING
   {AFTER }
   {{identifier-2}  [LINE ] }]
   {{integer     }  [LINES] }
   {[redacted]              }
```

PROCESS

A. Read entire statement

B. If the option "mnemonic-name" exists within the format, it should be eliminated. The operating system has commands for simulating this option.

C. Statement should be copied to the output error without "mnemonic-name"

OUTPUT

```
WRITE record-name
   [FROM identifier-1]
  [{BEFORE}  ADVANCING
   {AFTER }
   {identifier-2}  [LINE ]]
   {integer     }  [LINES]
```

NO: 2.3.2.10.2

INPUT	PROCESS	OUTPUT
WRITE record-name [FROM identifier] [INVALID KEY imperative-statement]	Read entire statement, then copy to output area	WRITE record-name [FROM identifier] [INVALID KEY imperative-statement]

NO: 2.3.3.1.1

INPUT	PROCESS	OUTPUT
ADD { identifier-1 / literal-1 } [identifier-2 / literal-2] . . . TO identifier-m [ROUNDED] [identifier-n [ROUNDED]] . . . [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area	ADD { identifier-1 / literal-1 } [identifier-2 / literal-2] . . . TO identifier-m [ROUNDED] [identifier-n [ROUNDED]] ... [ON SIZE ERROR imperative-statement]

NO: 2.3.3.1.2

INPUT	PROCESS	OUTPUT
ADD {identifier-1 / literal-1} {identifier-2 / literal-2} [identifier-3 / literal-3] . . . GIVING *identifier-m* [ROUNDED] [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area	ADD {identifier-1 / literal-1} {identifier-2 / literal-2} [identifier-3 / literal-3] . . . GIVING identifier-m [ROUNDED] [ON SIZE ERROR imperative-statement]

NO: 2.3.3.1.3

INPUT	PROCESS	OUTPUT
ADD {CORRESPONDING / CORR} identifier-1 TO identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area	ADD {CORRESPONDING / CORR} identifier-1 TO identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]

AD-A107 879   SAI COMSYSTEMS CORP MCLEAN VA   F/G 9/2
SOFTWARE PORTABILITY STUDY CONVERSION PROCEDURES.(U)
JUN 77 T DENIKE, A HOLLAND, T WARD, H DESAI   DAHC26-76-D-1004
UNCLASSIFIED   USACSC-AT-77-11   NL

3 OF 3
AD
A107879

END
DATE
FILMED
1-82
DTIC

NO: 2.3.3.2

INPUT	PROCESS	OUTPUT
COMPUTE identifier-1 [ROUNDED] [, identifier-2 [ROUNDED]] ... = arithmetic-expression [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area	COMPUTE identifier-1 [ROUNDED] [, identifier-2 [ROUNDED]] ... = arithmetic-expression [ON SIZE ERROR imperative-statement]

NO: 2.3.3.3.1

INPUT	PROCESS	OUTPUT
DIVIDE { identifier-1 / literal-1 } INTO identifier-2 [ROUNDED] [, identifier-3 [ROUNDED]] . . . [ON SIZE ERROR imperative-statement]	Read entire statement, then copy to output area	DIVIDE { identifier-1 / literal-1 } INTO identifier-2 [ROUNDED] [, identifier-3 [ROUNDED]] . . . [ON SIZE ERROR imperative-statement]

NO: 2.3.3.3.2

INPUT

```
DIVIDE {identifier-1} {INTO}
       {literal-1   } {BY  }

{identifier-2} GIVING
{literal-2   }

identifier-3 [ROUNDED]

[identifier-4 [ROUNDED]]

[ON SIZE ERROR

        imperative-statement]
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
DIVIDE {identifier-1} {INTO}
       {literal-1   } {BY  }

{identifier-2} GIVING
{literal-2   }

identifier-3 [ROUNDED]

[identifier-4 [ROUNDED]]

[ON SIZE ERROR

        imperative-statement]
```

NO: 2.3.3.4.1

INPUT

```
INSPECT identifier-1
    REPLACING

{ CHARACTERS BY { identifier-6 }
                { literal-4    }

  [ { BEFORE }  INITIAL  { identifier-7 } ]
  [ { AFTER  }           { literal-5    } ]

{ , { ALL     }  { identifier-5 }
    { LEADING }  { literal-3    }
    { FIRST   }

  BY { identifier-6 }  [ { BEFORE }
     { literal-4    }  [ { AFTER  }

  INITIAL { identifier-7 } ] } ... } ...
          { literal-5    } ]
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
INSPECT identifier-1
    REPLACING

{ CHARACTERS BY { identifier-6 }
                { literal-4    }

  [ { BEFORE }  INITIAL  { identifier-7 } ]
  [ { AFTER  }           { literal-5    } ]

{ , { ALL     }  { identifier-5 }
    { LEADING }  { literal-3    }
    { FIRST   }

  BY { identifier-6 }  [ { BEFORE }
     { literal-4    }  [ { AFTER  }

  INITIAL { identifier-7 } ] } ... } ...
          { literal-5    } ]
```

NO: 2.3.3.4.2

INPUT

```
INSPECT identifier-1
    TALLYING
      identifier-2  FOR
   { {ALL    }  {identifier-3} }
   { {LEADING}  {literal-1   } }
   {     CHARACTERS            }
   [ {BEFORE}       INITIAL
   [ {AFTER }
        {identifier-4}  ] ...
        {literal-2   }  ]

(CONTINUED ON NEXT PAGE)
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
INSPECT identifier-1
   TALLYING
     identifier-2  FOR
 { {ALL    }  {identifier-3} }
 { {LEADING}  {literal-1   } }
 {     CHARACTERS            }
 [ {BEFORE}       INITIAL
 [ {AFTER }
     {identifier-4}  ] ...
     {literal-2   }  ]
```

INPUT	PROCESS	OUTPUT

REPLACING
{CHARACTERS BY {identifier-6 / literal-4}
[{BEFORE / AFTER} INITIAL {identifier-7 / literal-4}]
{ALL / LEADING / FIRST} {identifier-5 / literal-3}
BY {identifier-6 / literal-4}
[{BEFORE / AFTER} INITIAL {identifier-7 / literal-5}] ...
	Read entire statement, then copy to output area	REPLACING
		CHARACTERS BY {identifier-6 / literal-4}
		[{BEFORE / AFTER} INITIAL {identifier-7 / literal-4}]
		{ALL / LEADING / FIRST} {identifier-5 / literal-3}
		BY {identifier-6 / literal-4}
		[{BEFORE / AFTER} INITIAL {identifier-7 / literal-5}] ...

NO: 2.3.3.3.3

INPUT

```
DIVIDE  { identifier-1 }  { INTO }
        { literal-1    }  { BY   }

{ identifier-2 }  GIVING
{ literal-2    }

identifier-3 [ ROUNDED ]

REMAINDER identifier-4

[ ON SIZE ERROR
      imperative-statement ]
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
DIVIDE  { identifier-1 }  { INTO }
        { literal-1    }  { BY   }

{ identifier-2 }  GIVING
{ literal-2    }

identifier-3 [ ROUNDED ]

REMAINDER identifier-4

[ ON SIZE ERROR
      imperative-statement ]
```

NO: 2.3.3.5.1

INPUT	PROCESS	OUTPUT
<u>MULTIPLY</u> {identifier-1 / literal-1} <u>BY</u> identifier-2 [<u>ROUNDED</u>] [identifier-3 [<u>ROUNDED</u>]] ... [ON <u>SIZE</u> <u>ERROR</u> imperative-statement]	Read entire statement, then copy to output area	<u>MULTIPLY</u> {identifier-1 / literal-1} <u>BY</u> identifier-2 [<u>ROUNDED</u>] [identifier-3 [<u>ROUNDED</u>]] ... [ON <u>SIZE</u> <u>ERROR</u> imperative-statement]

NO: 2.3.3.5.2

INPUT

```
MULTIPLY  { identifier-1 }  BY
          { literal-1    }

  { identifier-2 }  GIVING
  { literal-2    }

  identifier-3  [ROUNDED]

  [, identifier-4

     [ROUNDED] ] . . .

  [ON SIZE ERROR
       imperative-statement]
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
MULTIPLY  { identifier-1 }  BY
          { literal-1    }

  { identifier-2 }  GIVING
  { literal-2    }

  Identifier-3  [ROUNDED]

  [, identifier-4

     [ROUNDED] ] . . .

  [ON SIZE ERROR
       imperative-statement]
```

NO: **2.3.3.6.1**

INPUT	PROCESS	OUTPUT
<u>SUBTRACT</u> {identifier-1 | literal-1} [, identifier-2 | , literal-2] <u>FROM</u> identifier-m [<u>ROUNDED</u>] [, identifier-n [<u>ROUNDED</u>]] ... , ON <u>SIZE</u> <u>ERROR</u> imperative-statement]	Read entire statement and copy to output area	<u>SUBTRACT</u> {identifier-1 | literal-1} [, identifier-2 | , literal-2] ... <u>FROM</u> identifier-m [<u>ROUNDED</u>] [, identifier-n [<u>ROUNDED</u>]] [, ON <u>SIZE</u> <u>ERROR</u> imperative-statement]

NO: 2.3.3.6.2

INPUT

```
SUBTRACT { identifier-1 }
         { literal-1    }

  [ , identifier-2 ]
  [ , literal-2    ] . . .

  FROM { identifier-m }
       { literal-m    }

  GIVING identifier-m
      [ROUNDED]
  [ , identifier-o
      [ROUNDED] ] . . .

  [ , ON SIZE ERROR
      imperative-statement ]
```

PROCESS

Read entire statement and copy to output area

OUTPUT

```
SUBTRACT { identifier-1 }
         { literal-1    }

  [ identifier-2 ]
  [ literal-2    ] . . .

  FROM { identifier-m }
       { literal-m    }

  GIVING identifier-n
      [ROUNDED]
  [ identifier-o
      [ROUNDED] ] . . .

  [ , ON SIZE ERROR
      imperative-statement ]
```

NO: 2.3.3.6.3

INPUT	PROCESS	OUTPUT
SUBTRACT { CORRESPONDING / CORR } identifier-1 FROM identifier-2 [ROUNDED] [;ON SIZE ERROR imperative-statement]	Read entire statement and copy to output area	SUBTRACT { CORRESPONDING / CORR } identifier-1 FROM identifier-2 [ROUNDED] [ON SIZE ERROR imperative-statement]

NO: 2.3.4.1.1

INPUT	PROCESS	OUTPUT
$\underline{\text{MOVE}} \begin{Bmatrix} \text{identifier-1} \\ \text{literal} \end{Bmatrix} \underline{\text{TO}}$ identifier-2 $\left[\text{, identifier-3} \right] \ldots$	Read entire statement, then copy to output area	$\underline{\text{MOVE}} \begin{Bmatrix} \text{identifier-1} \\ \text{literal} \end{Bmatrix} \underline{\text{TO}}$ identifier-2 $\left[\text{, identifier-3} \right] \ldots$

NO: 2.3.4.1.2

INPUT	PROCESS	OUTPUT
$\underline{\text{MOVE}} \left\{ \begin{array}{l} \underline{\text{CORRESPONDING}} \\ \underline{\text{CORR}} \end{array} \right\}$ identifier-1 $\underline{\text{TO}}$ identifier-2	Read entire statement, then copy to output area	$\underline{\text{MOVE}} \left\{ \begin{array}{l} \underline{\text{CORRESPONDING}} \\ \underline{\text{CORR}} \end{array} \right\}$ identifier-1 $\underline{\text{TO}}$ identifier-2

NO: 2.3.4.2.1

INPUT

```
SEARCH identifier-1

   [VARYING {identifier-2 }]
            {index-name-1 }

   [AT END imperative-
        statement-1 ]

    WHEN condition-1
        {imperative-statement}
        {NEXT SENTENCE       }

   [WHEN condition-2
        {imperative-statement-3}]
        {NEXT SENTENCE         }

             . . .
```

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
SEARCH identifier-1

   [VARYING {identifier-2 }]
            {index-name-1 }

   [AT END imperative-
        statement-1 ]

    WHEN condition-1
        {imperative-statement}
        {NEXT SENTENCE       }

   [WHEN condition-2
        {imperative-statement-3}]
        {NEXT SENTENCE         }

             . . .
```

NO: 2.3.4.2.2

INPUT

```
SEARCH ALL  identifier-1
    [AT END imperative-
        statement-1]

        data-      {IS EQUAL TO}
        name-1     {IS =       }

WHEN      {identifier-3     }
          {literal-1        }
          {arithmetic-exp-1 }

        condition-name-1

    [   data-      {IS EQUAL TO}
        name-2     {IS =       }

    AND   {identifier-4     }
          {literal-2        }
          {arithmetic-exp-2 }

        condition-name-2       ]

    {imperative-statement-2}
    {NEXT SENTENCE         }
```

PROCESS

Read entire statement, then copy to output area.

OUTPUT

```
SEARCH ALL identifier-1
    [AT END imperative-
     statement-1]

        data-      {IS EQUAL TO}
        name-1     {IS =       }

WHEN      {identifier-3     }
          {literal-1        }
          {arithmetic-exp-1 }

        condition-name-1

    [   data-      {IS EQUAL TO}
        name-2     {IS =       }

    AND   {identifier-4     }
          {literal-2        }
          {arithmetic-exp-2 }

        condition-name-2       ]

    {imperative-statement-2}
    {NEXT SENTENCE         }
```

NO: 2.3.4.3.1

INPUT

```
SET  { identifier-1 }
     { index-name-1 }

   [, identifier-2] ...
   [, index-name-2] ...

TO

   { identifier-3 }
   { index-name-3 }
   { integer-1    }
```

PROCESS

Read entire statement and copy to output area

OUTPUT

```
SET  { identifier-1 }
     { index-name-1 }

   [identifier-2] ...
   [index-name-2] ...

TO

   { identifier-3 }
   { index-name-3 }
   { integer-1    }
```

NO: 2.3.4.3.2

INPUT	PROCESS	OUTPUT
SET index-name-4 [, index-name-5] ... $\left\{\begin{array}{l}\text{UP BY}\\\text{DOWN BY}\end{array}\right\}\left\{\begin{array}{l}\text{identifier-4}\\\text{integer-2}\end{array}\right\}$	Read entire statements and copy to output area	SET index-name-4 [index-name-5] ... $\left\{\begin{array}{l}\text{UP BY}\\\text{DOWN BY}\end{array}\right\}\left\{\begin{array}{l}\text{identifier-4}\\\text{integer-2}\end{array}\right\}$

NO: 2.3.4.4

INPUT

```
STRING   identifier-1
         literal-1

  [, identifier-2]
  [, literal-2   ]  . . .

DELIMITED BY  identifier-3
              literal-3
              SIZE

[, identifier-4] [, identifier-5]
[, literal-4   ] [, literal-5   ]

      . . .

DELIMITED BY  identifier-6
              literal-6
              SIZE

INTO identifier-7

[WITH POINTER identifier-8]

[ON OVERFLOW
    imperative-statement]
```

PROCESS

Read entire statements and copy to output area

OUTPUT

```
STRING  {identifier-1}
        {literal-1   }

  [identifier-2]
  [literal-2   ]  . . .

DELIMITED BY {identifier-3}
             {literal-3   }
             {SIZE        }

[{identifier-4} [identifier-5]
 {literal-4   } [literal-5   ]

      . . .

DELIMITED BY {identifier-6}
             {literal-6   }
             {SIZE        }]

      . . .

INTO identifier-7
[WITH POINTER identifier-8]
[ON OVERFLOW
    imperative-statement]
```

NO: 2.3.4.5

INPUT

```
UNSTRING identifier-1

  [DELIMITED BY [ALL]
      {identifier-2}
      {literal-1   }
   [, OR [ALL] {identifier-3}]
               {literal-2   }
          ...] INTO identifier-4
[, DELIMITER IN identifier-5]
[, COUNT IN identifier-6]
[, identifier-7
[, DELIMITER IN identifier-8]
[, COUNT IN identifier-9]]  ...
```

(CONTINUED ON NEXT PAGE)

PROCESS

Read entire statement, then copy to output area

OUTPUT

```
UNSTRING identifier-1

  [DELIMITED BY [ALL]
      {identifier-2}
      {literal-1   }
   [, OR [ALL] {identifier-3}]
               {literal-2   }
       ...] INTO identifier-4
[, DELIMITER IN identifier-3]
[, COUNT IN identifier-6]
[, identifier-7
[, DELIMITER IN identifier-8]
[, COUNT IN identifier-9]]  ...
```

INPUT	PROCESS	OUTPUT
[WITH POINTER identifier-10] [TALLYING IN identifier-11] [ON OVERFLOW imperative-statement]		[WITH POINTER identifier-10] [TALLYING IN identifier-11] [ON OVERFLOW imperative-statement]

NO: 2.3.5.1

INPUT	PROCESS	OUTPUT
$\underline{\text{IF}}\ \text{condition} \begin{Bmatrix} \text{statement-1} \\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}} \end{Bmatrix}$ $\begin{Bmatrix} \underline{\text{ELSE}}\ \text{statement-2} \\ \underline{\text{ELSE}}\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}} \end{Bmatrix}$	Read entire statement, then copy to output area	$\underline{\text{IF}}\ \text{condition} \begin{Bmatrix} \text{statement-1} \\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}} \end{Bmatrix}$ $\begin{Bmatrix} \underline{\text{ELSE}}\ \text{statement-2} \\ \underline{\text{ELSE}}\ \underline{\text{NEXT}}\ \underline{\text{SENTENCE}} \end{Bmatrix}$

FILMED
8